Proceedings of the 2002 IEEE
Workshop on Information Assurance and Security
United States Military Academy, West Point, NY, 17–19 June 2002

Defending Against a large scale Denial-of-Service Attack

Department of Computer Science
Columbia University
New York, NY 10027

Suhail Mohiuddin, Shlomo Hershkop, Rahul Bhan,and Sal Stolfo
{sm1151,shlomo,rb839,sal}@cs.columbia.edu

Abstract—

The number of Denial-of-Service (DOS) attacks has been increasing alarmingly over the past few years. Most often, corporate web sites with heavy traffic flows are targeted. In general, a specific service is the target of the DOS attack, but more often than not, the result of a DOS attack will be the effective disruption both of the service and host computer.

This paper attempts to develop a new and efficient technique for the detection and alleviation of DOS attacks. We present a new block based technique that may provide significant reduction of overhead in processing and analyzing traffic data. We propose a distributed system that collects diagnostic data from sensors across the mirror servers and computes the condition of the network. This data is mined in real time to generate traffic models. Incoming traffic is matched against traffic models to check for anomalies and corrective measurements are taken as needed.

I. Introduction

Denial-of-Service (DOS) attacks have become increasingly common and frequent over the years [1], [2]. From 1989 to 1995 the number of DOS attacks doubled each year [3]. Last year, Microsoft's official web site was down for more than a day due to this type of attack. A DOS attack may be broadly defined as an explicit attempt to prevent legitimate consumers of a service from using the service [2].

In the most common DOS attack, an attacker floods the victim's network by transmitting millions of garbage packets or by issuing illegitimate requests to consume the web server's resources. This method of issuing fake requests can be classified into two types; SYN attacks and HTTP request attacks. In this paper we will use the term 'request' interchangeably with a SYN request and a HTTP request, however we will make the distinction between them whenever needed.

Issuing fake requests can lead to web servers overloading, destabilizing, and possibly crashing or re-booting, leading to even greater vulnerabilities. In addition, legitimate requests arriving at the server during this time are unable be processed. This can result in large losses on the part of the host computer's owners. In today's e-commerce environment, users have a low tolerance for web site delay or failure and will click to another site if the first is unavailable. Moreover, attacks like DOS are very disturbing, since an individual with nefarious aspirations may arbitrarily cause havoc to corporate servers. DOS scripts and tools are freely available at a number of locations on the Internet, thereby making it very difficult and time consuming for corporations to identify and prosecute an attacker.

The Internet was originally designed to work on mutual trust between machines. This paradigm assisted the growth and spread of the Internet, but ironically, it is also this trust that assists attackers in covering their tracks. Accountability was never a factor while designing the Internet [4].

A coordinated attack, commonly referred to as a Distributed Denial of Service (DDOS) attack, consists of a synchronized attack by multiple machines [5]. The attacker gains illegal control of multiple machines and launches a simultaneous attack on a victim system. In February 2000, This type of attack was successfully used against large commercial sites such as Yahoo! and e-bay.com among others. A detailed look at the inner workings of DDOS attacks can be found in [6].

There are more sophisticated DOS attacks, with attackers using fake source addresses in the IP headers. Other procedures in attacks are used to bypass filtering rules that some web servers and routers employ. All of this makes handling DOS attacks extremely difficult. Current solutions include ingress/egress filtering implemented at the ISP edge routers, special Operating Systems [7], and DOS proof network appliances.

RFC 2827 [8] provides guidelines to the Internet community on how to defeat IP-Address spoofing. The most important guideline suggests that service providers use a simple ingress filtering method. Each edge router should only allow traffic with a source IP matching its advertised subnet to pass on to the core network. It is suggested that if this method were widely implemented, it would virtually eliminate IP spoofing. However, it mentions that ISP's would need to discontinue some of their services in order to accommodate such filtering. For example, Mobile IP [9] would have to implement reverse tunneling and would result in a significant degradation of system perfor-

ISBN 0-7803-9814-9/$10.00

mance. Some ISP's have already begun to conform to these guidelines. However, the current number of ISP's not using ingress filtering is still substantial. In addition, some types of DOS attacks, like ping floods, can be stopped by disabling the requisite ports. But in most cases, prevention of a DOS attack at the ISP level entails shutting down the affected party's network link.

DOS proof hardware appliances include DDos Enforcer by Mazu Networks [10] and Peakflow by Arbor Networks [11]. Mazu Network's approach is to implement packet filtering with an anomaly detection engine, which issues alerts based on bandwidth and suspicious thresholds. The anomaly engine is trained on the network to seed a "normal" profile for bandwidth and packets, and when the profile is violated, alerts are issued. In addition, filters can be set up by the user to effectively clean up bad traffic. The problem with this approach is the thresholds depend on the data seen during training. New services or high demand for a popular service will trigger the alert mechanism. In addition, the features of bandwidth and network levels do not adequately model the complexities of dynamic computer network flows.

We suggest a novel approach for detection and response toward a DOS attack. Our network topology consists of sensor agents called DCA (Data Collection Agents) located at the edges and managing agents called DFA (Data Fusion Agents) located inside the network. The DFA's generate traffic models based on current network traffic and the DCA's use these models to detect anomalous traffic. If traffic is deemed anomalous, then the DCA can either drop it or load balance it to another low priority server.

For attack detection, we built upon preliminary research at Columbia University on data-mining methods applied to intrusion detection. The idea is, by using data mining, knowledge of known traffic patterns and attacks can be generalized to detect known and unknown attacks. Data mining is a good candidate in such situations, where large volumes of traffic need to be combed for patterns and these patterns used to detect anomalous traffic instances [12].

The paper is structured as Follows: Section 2 gives a description of DOS attacks, Section 3 describes our proposed high-level system architecture, and sections 4-7 explains the operation of our system. We describes the design choices and trade-offs. We suggest a comparative-based scheme using a data-mining approach of profiling traffic. A profile is built up over network traffic; it may include various features such as traffic load, IP addresses, CPU load, memory used and low level network header information. We generate traffic profiles using training data and then test these profiles on test data using k-fold cross validation. In addition, experiments were run to adjust the type of block and to test what percentage of the block had to be analyzed at the expense of throughput.

II. Background

The number of malicious programs intended to carry out denial of service attacks are increasing both in number, availability, and sophistication. Some of them are designed to take advantage of common vulnerabilities of different operating systems. For example, many operating systems can crash when receiving malformed network packets. Most of the DOS programs are 'brute force', designed for flooding and bogging down the system by sheer force. Vulnerability attacks can be remedied by applying suitable patches to the intended victim's operating system.

Programs like CrashIIS send a malformed GET request, "GET ../..", to a Windows NT machine running certain versions of IIS causing IIS to crash. Such attacks only affect older operating systems, as new ones are installed with security features that either prevent such malicious requests from being serviced or provide operating system support to prevent crashes. In general, the vulnerability window of patches is quite large. The flooding attacks, however, cannot be easily identified nor be easily prevented, as they send out legitimate requests in 'illegitimate' numbers. The 'illegitimate' amount is hard to quantify, as high amounts of traffic can be attributed to a legitimate user-wide interest in a resource, fetching the latest headline of a breaking news story, for example. Some of the common programs that are used for flooding attacks are described below.

Mailbomb

A Mailbomb is an attack in which the attacker sends many messages to a server, overflowing that server's mail queue and possibly causing system failure. A typical attack will send 10,000 one-kilobyte messages (10 megabytes of total data) to a single user. This attack is more of a nuisance for a particular user than a real threat to the overall security of a server. Although, with a simple adoption, the flooding attack can quickly become a vulnerability attack.

SYN flood

A SYN Flood is a denial of service attack to which every TCP/IP implementation is vulnerable, to some degree. Each half-open TCP connection made to a machine causes the 'tcpd' server to add a record to the data structure that stores information describing all pending connections. This data structure is of finite size, and it can be made to overflow by intentionally creating too many partially open connections. The connections data structure on the victim server system will eventually fill and the system will be unable to accept any new incoming connections until the table is emptied out. Normally there is a time out associated with a pending connection, so the half-open connections will eventually expire, and the server system will recover. However, the attacking system can simply continue sending IP-spoofed packets requesting new connections faster than the victim system can expire the pending connections. In

ISBN 0-7803-9814-9/$10.00

some cases, the system may exhaust its memory, crash, or be rendered otherwise inoperative.

Smurf

In the "Smurf" attack, attackers use ICMP echo request packets directed to IP broadcast addresses from remote locations to create a denial-of-service attack. There are three parties in these attacks: the attacker, the intermediary, and the victim (note that the intermediary can also be a victim). The attacker sends ICMP 'echo request' packets to the broadcast address xxx.xxx.xxx.255 of many subnets with the source address spoofed to be that of the intended victim. Any machines that are listening on these subnets will respond by sending ICMP 'echo reply' packets to the victim. The smurf attack is effective because the attacker is able to use broadcast addresses to amplify what would otherwise be a rather innocuous ping flood. In the best case (from an attacker's point of view), the attacker can flood a victim with a volume of packets 255 times as great in magnitude as the attacker would be able to achieve without such amplification. Because there can be as many as 255 machines on an Ethernet segment. Usually, the attacker sends a stream of ICMP 'ECHO' requests to the broadcast address of many subnets, resulting in a large, continuous stream of 'ECHO' replies that flood the victim. The Smurf attack can be identified by an intrusion detection system that notices that there are a large number of 'echo replies' being sent to a particular victim machine from many different addresses, but no 'echo requests' originating from the victim machine.

TCP Reset

TCP Reset is a denial of service attack that disrupts TCP connections made to the victim machine. That is, the attacker listens (on a local or wide-area network) for TCP connections to the victim, and sends a spoofed TCP RESET packet to the victim, thus causing the victim to inadvertently terminate its own TCP connection.

UDPstorm

A UDPstorm is a DOS attack that causes network congestion and slowdown. When a connection is established between two UDP services, these two services can produce a very high number of packets that can lead to a denial of service on the machine(s) where the services are offered. Anyone with network connectivity can launch an attack; no account access is needed. For example, by connecting a host's chargen service to the echo service on the same or another machine, all affected machines may be effectively taken out of service because of the excessively high number of packets produced. First, the attacker forges a single packet that has been spoofed to look like it is coming from the echo port on the first victim machine and sends it to the second victim. The echo service blindly responds to any request it receives by simply echoing the data of the request back to the machine and port that sent the echo request, so when the victim receives this spoofed packet it sends a response to the echo port of the first victim. This first victim responds in like kind, and the loop of traffic continues until it is stopped by intervention from an external source. This attack can be identified in two ways. First, the single packet that initiates the attack can be recognized because it is a packet originating from outside the network that has been spoofed to appear as if it is coming from a machine inside the network. Second, once the loop of network traffic has been initiated, an intrusion detection system that can see network traffic on the inside of the network can note that traffic is being sent from the chargen or echo port of one machine to the chargen or echo port of another.

III. System Architecture

We assume that the network consists of distributed web servers serving the 'outside' world. Each web-server mirrors the same content. In our scheme, a Data Fusion Agent (DFA) is located in a private network, connected to all the Data Collection Agents (DCAs) located at the web servers. It is recommended to have a separate network that connects all these DCA's to the central DFA. This becomes important during a DOS attack when massive congestion occurs on the public network near the web-servers. We also assume that, in general, a full-fledged DOS attack or a Distributed DOS attack would target most of the mirror web-servers. This is a valid assumption because a localized DOS attack may not harm the total functioning of the web site.

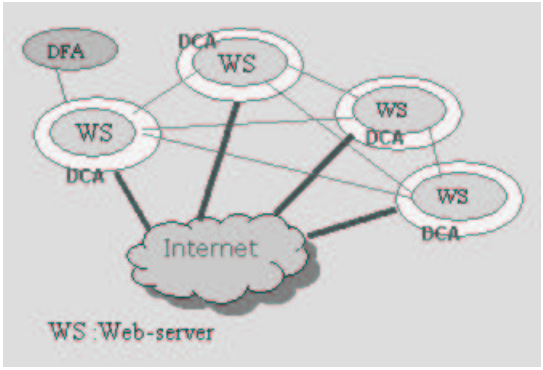

Fig. 1. The network topology. Note that the DCA's are software wrappers around the web-servers. The DFA randomly moves around, maintaining a network-wide view of the whole system

The DCA's form a layer between the web-server and the Internet. Note that this is a one-way layer; although the DCA intercepts the incoming request, it does not have any role once the request reaches the web-server. The web-server services the request directly through the Internet, bypassing the DCA. There are two primary roles of the

DCA. The first is to analyze the traffic data by sending reports to the DFA. The DFA collects traffic data from the DCA and uses machine learning algorithms to generate traffic models. The traffic models are sent to the DCA. The second major role of DCA is to use these traffic models and act like a model analyzer. It operates on blocks of incoming requests using the traffic model provided by the DFA and deciding whether the current block of incoming traffic is anomalous or not. The decision to drop a block or not depends on the mode in which the DCA is operating. The DCA decides immediately about the nature of the block, it does not need to wait for any information from the DFA. This factor directly enhances the speed and scalability of the system. Based on previous work done by NRL [13] we incorporated the ideas of randomly roving agents into the system.

Because of the nature of a DOS attack, where service is interrupted or lost from a host, we move away from a fixed structure in favor of a decentralized coordinating center. The DFA can actually start off on a randomly chosen DCA host. Random probes are sent out to the other DCA's to coordinate information. As soon as a DFA is lost, one of the random probes will designate itself as a new DFA and send out new probes. In the event of two DFAs being set up, a rule-based approach will be used to decide between the two. It is interesting to note the number of DFA's required to collect unique data over a M-node network, hopping from node to node. Recent NRL work [13] on random roving agents lets us evaluate the number of roving agents needed to retrieve unique information from M-nodes. If we consider K agents, moving across a network of M nodes, with each one picking up n unique nodes, the average number of nodes visited is given by F = M(1 - (1 -(n/M))K) . In particular, for a network with 1000 nodes and each agent visiting 20 nodes, about 35 DFA's will be required to get a mean of 500 unique node traversal.

In addition, load balancing the incoming requests is also an important aspect of dealing with DOS attacks. In an environment where content is distributed across mirrored servers, the DCA/DFA configuration can try to alleviate suspicious network loads by first attempting to serve requests by passing it out amongst themselves. In this way, the amount of traffic per host is reduced, allowing DCA's to analyze the blocks of traffic. The load balancing mechanisms are programmed into the DFA, allowing it to spawn new mirrors and DCA's as needed.

IV. Design

The Data Collection Agents (DCA's) are located between the Internet and the server. They run in multiple severity modes, depending on status messages from the DFA. Their primary function is to act as traffic model analyzers. In passive mode they allow all connections to go through while simultaneously generating current traffic profiles and sending summary profiles to the DFA. If the DFA suspects an attack, it switches to 'suspect' mode, can request more data, and do a more thorough check of connections passing through. If suspicious traffic persists, the DCA then switches to a more vigilant (active) mode and starts analyzing connection blocks before allowing them to connect to the server. If the DFA confirms an attack, then the DCA can start to drop batches of connections if they are deemed anomalous.

The Data Fusion Agent (DFA) is located in the internal private network of an organization and has an overall, network-wide view of the whole system. Its primary function is to act as a traffic model generator. It receives connection records from the DCA's and uses them to update the current network traffic model. The model is algorithmic dependent, but represents the information needed by the DFA to analyze the network flow. The DFA also instructs and coordinates the DCA's which are distributed over the network. As mentioned previously, the DFA uses machine learning and data mining algorithms for generating the models. The data sets used for training are extensive enough to mimic real network activity. In addition, real-time analysis has to be observed, which means we carefully select the features in order to obtain good response times. For example, we can use information extracted from the SYN packet to the FIN packet and all requests that lie in between. On a higher level, we need to take into account the statistics of the traffic. These statistics include traffic volume, type of traffic and pattern of requests.

The concept of using modes as a control for the operation of DCA is introduced primarily to save computation expense and memory, during 'normal times'.

Since the DFA has a network wide view, it can also act as a load balancer. It has the knowledge of which DCA's are being overloaded with traffic and which DCA's have blocks of data that can be considered suspect. It can redirect this traffic to other DCA's that may be idle, or it can direct the anomalous traffic to 'honeypot' machines for logging and future investigation. As a converse, it can also redirect 'good' traffic from servers that are facing the brunt of the attack.

To enhance efficiency, we propose using block-based selection of requests. Using this technique, we classify incoming requests in blocks of time. Each period of time will consist of a block. All requests arriving during this time will be collected in a block. We would randomly sample X percent of the requests for analyzing the traffic. In the normal case, this number, X, would be relatively low, thus minimizing the computational load on the DCA and thereby reducing overhead. These savings become significant in high traffic web-servers. It can be shown [9] that a relatively small number of random samples in a block can depict the state of the traffic quite accurately.

The DCA implements a coarse analysis of the randomly

ISBN 0-7803-9814-9/$10.00

selected traffic. The DFA will carry out a more extensive analysis using more computational intensive algorithms.

A. Passive Mode

In a passive mode, the web-server operates freely. The DCA compares incoming traffic to the current traffic profile, previously generated by the DFA. Anomalous traffic may be allowed to go through at this stage, although this is noted and sent to DFA by the DCA. The traffic profile may include features such as source IP address, port number, volume of requests, number of request per source IP, CPU load, memory utilization etc.

B. Suspect Mode

In case the DFA generates a model suggesting that a possible attack is underway, it can instruct the DCA's to increase vigilance. Thus, the DCA's no longer allow the traffic to flow freely. The DCA selects a larger percentage of connections for analysis, and may occasionally reject anomalous blocks of traffic but still allows most traffic through. This situation could arise if the network is going through a series of scans, which could be a precursor to an attack.

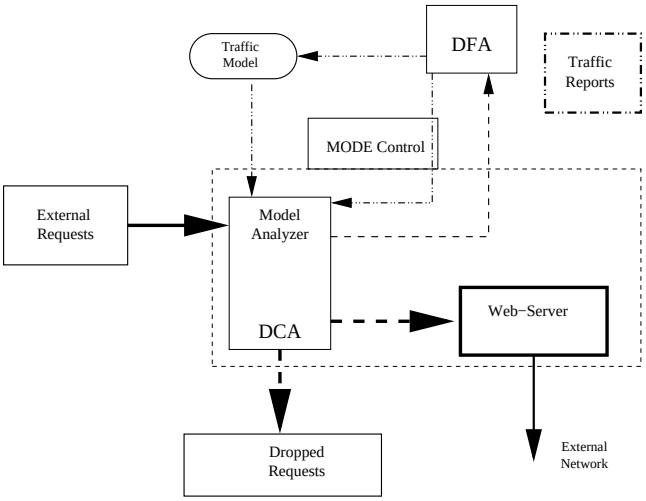

Fig. 2. Schematic showing the operation of DCA and its mode control. The DCA is continuously sending traffic reports to the DFA and getting traffic models from it. The DFA also determines the mode in which the DCA operates

C. Active Mode

This mode is activated at the DCA when the current traffic model generated by the DFA suggests, with high certainty, that an attack is underway. This is the highest level of alertness. This mode implements the severest measures, filtering out suspected IP's and connections, to reduce the block sizes to a minimum. It filters out suspected IP addresses by employing a probabilistic method. A simple rule would be to only allow an IP (or a subnet) which has been 'behaving' well for long periods of time. By well-behaving we mean this IP/subnet has been observed to follow a normal model in the last few blocks. Normal models of usage include no incomplete resource requests within the last few blocks. As a consequence of this measure, the good well-behaved consumers of a resource will be served with greater priority at the time of a heavy congestion due to an attack. This trade-off will mean any new incoming requests from a new IP address at the time of attack might not be served. We believe this is a reasonable trade-off to make. A case is made in [14] that precious resources are wasted in processing incoming packets during the DOS attack. We alleviate some of the pressure by dropping the block of connections if we are in Active mode and the block contains many anomalies or suspicious connections. The TCP protocol will allow innocent victims to recover gracefully and re-attempt the request within a reasonable time frame.

D. Load Balancing

A load-balancing scheme can also be incorporated easily into our system with the DFA instructing DCA's to transfer excess loads from a heavily loaded machine to another server. The system-wide view available to the DFA can allow it to manage network resources with ease. Load Balancing the system can be implemented as follows: Models of DOS attacks can be built so that if a large amount of requests are made, but they do not match the current DOS models, new server and DCA agents can be spawned or mirrored and the situation monitored. If the system stabilizes with a larger allocation of resources, we can take this as an indication of increase in legitimate demand. If the situation continues to deteriorate, we can assume it is a new type of DOS attack and take measure to correct the situation. In addition, load balancing allows us to decrease network resources in conjunction to a decrease in demand.

E. Advantages

Using our block based sampling method, the system can provide high efficiency throughput. This is especially true during normal network activity. The concept of DCA's operating in modes, enables a low overhead in normal times, thus reducing loads. The system provides fast request handling, because the decision to pass a block or not primarily rests with the DCA. The system is scalable, since you can always scale the number of DCA's and DFA's for a larger network. The DCA's provide a protective layer between the Internet and the web-servers, so in the event of a heavy DOS attack, the web-server will not crash.

V. Feature extraction

We built a rule-based learner similar to what is described in [15] to choose the relevant features. An inductive algorithm learns a model of an attack using training data. We

use the following features in training.

1. Is there a sudden increase of traffic at this node?
2. Are all the nodes having a surge in traffic?
3. Is any single source getting more than reasonable share of resources?
4. Are the requests and connections valid?
5. Is it an expected increase of traffic? (To actually decide this feature, other information across multiple DCAs must be evaluated).
6. Are hosts reachable?

The classifier we incorporated into the system consists of a naive Bayes classifier [15]. A Bayes classifier computes the likelihood that a block is 'faulty', given the features of the block. We assumed that there were common traffic patterns in DOS attacks that differentiated them from normal traffic and other types of attack.

For instance, if traffic suddenly starts to surge with repeated IP addresses, repeated unfinished requests or from unreachable hosts, the Bayes algorithm starts to label the traffic blocks as suspect and alerts the DCA. The naive Bayes algorithm computes the probability that a given feature was malicious, and the probability that a given feature was benign by computing statistics over training data. The data used for this paper was from the DARPA 1999 data sets [16].

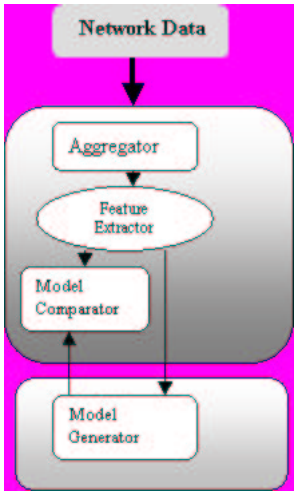

Fig. 3. Details of functioning of the DCA and DFA. The aggregator collects the requests to form blocks. The DFA is simultaneously connected to a number of DCAs (not shown).

Like any inductive learner, no a priori assumptions are made regarding the final concept. The learner makes its primary assumption that the data seen in training is similar to the data that will be seen in the future (i.e. they will come from the same distribution). This is evaluated by having labeled data, training over some of the data, and testing our models over the rest.

A. Tests

The initial aspect of our design involves the generation of traffic models. These models were used to evaluate the incoming connections. The 1999 DARPA data set [16] contains extensive data of normal and attack traffic patterns documented over a period of a few weeks. We used this data set for our model generation, 70% for training, and the remaining 30% for testing.

To get a better understanding of the labeled data set, we broke it up into per minute blocks of connections, and then built a probability table based on these blocks. This table contained information regarding the percentage of HTTP requests per block, the percentage of FTP requests per block etc. We used the entire block for generating this data (as this is a training data set). Since the data is labeled, we classified each row of the table as having contributed to normal or attack traffic. Our models are essentially a boolean expression of the columns from the probability table. For example x% of HTTP requests AND m% of FTP requests AND Source IP 'm' = True, where True and False signify anomalous and normal traffic respectively. 'x' and 'm' are observed values corresponding to a set threshold. We then use these rules to check new unlabeled data. If a significant deviation from the model is observed, then we can classify the data as anomalous. This would lead to increasing the surveillance mode of the DCA, which would then lead to a per connection analysis of a random k% of the block. This k% can be increased incrementally, so an increasing number of connections are analyzed. Connections deemed as anomalous are then dropped or load balanced to other machines.

VI. Algorithms

In this section we describe the Rule-based learner algorithm that was implemented. The algorithm was used on a set of features that were extracted beforehand from the DARPA data. These features included:

1. Number of HTTP requests
2. Number of FTP requests
3. Number of SNMP requests
4. Number of 'other' requests
5. Maximum number of requests by an IP

These were all based on 'per block' where each block was one minute of time.

A. Rule Based Learner

The algorithm we used is a rule-based inductive learner that builds a set of rules over the above mentioned features. We associated threshold values over all the features obtained during training, and if a certain threshold was breached then an anomaly was recorded. An example of a simple rule is displayed in Table I.

Similarly, we wrote other sets of rules that try to correlate the number of maximum requests with the number of

TABLE I
SIMPLE RULE

HTTP > Threshold	No
SNMP > Threshold	Yes
Max Requests > Threshold	Yes
Malicious?	Yes

HTTP connections to see if a TCP SYN attack is underway. This is a simplistic mechanism and has a high false positive rate that can be adjusted, but it offers extremely fast computation. It is a trade-off between choosing complex rules over a wide set of features, or choosing simple rules over features and having fast computational time. For our experiments we chose the latter.

VII. EXPERIMENTS

We conducted our experiments using DARPA's 1999 Lincoln Labs data set [16], [17]. This data set is a publicly available and recognized data set for testing IDS's. There is a concern that simulated data might nor accurately portray real-world networks, thus results might not extend to real-world networks. However, we address this concern in the future work section.

The DARPA data consisted of tcpdump data from a simulated Air Force network. Within the data, there appear a number of Denial-of-Service attacks, which included attacks like DOSNuke, selfping, syslog, mailbomb, and ping-of-death [16]. Most of them, like DOSNUKE and selfping, exploit system vulnerabilities and lead to system crashes. These attacks can be prevented by applying proper patches to the system. Other attacks like mailbomb and ping-of-death flood the system and cause it to slow down or crash. We concerned ourselves only with identifying flooding attacks. To evaluate our system we were interested in several quantities:

1. True Positives (TP), the number of malicious blocks classified as malicious blocks.
2. True Negatives (TN), the number of benign blocks classified as benign blocks.
3. False Positives (FP), the number of benign blocks classified as malicious blocks.
4. False Negatives (FN), the number of malicious blocks classified as benign blocks.

We were interested in the detection rate of the classifier. In our case, this was the percentage of the total malicious blocks labeled malicious. We were also interested in the false positive rate. This was the percentage of benign blocks that were labeled as malicious, also called false alarms.

- The detection rate is defined as TP/(TP+FN)
- False Positive rate a FP/(TN+FP)
- Overall Accuracy as (TP+TN)/(TP+TN+FP+FN)

The results of experiments are presented in Table II and in Figures 4 and 5. These are the results of classification using a rule-based learner. The idea is to classify bad blocks based on their attributes and using training data to characterize traffic. DARPA's 1999 evaluation data was used for detection and classification of flooding DOS attacks.

TABLE II
CLASSIFICATION

Profile Type	Rule-Based Learner
True Positive	62.5%
True Negative	99.7%
False Positive	37.5%
False Negative	37.5%
Overall Accuracy	68.38%

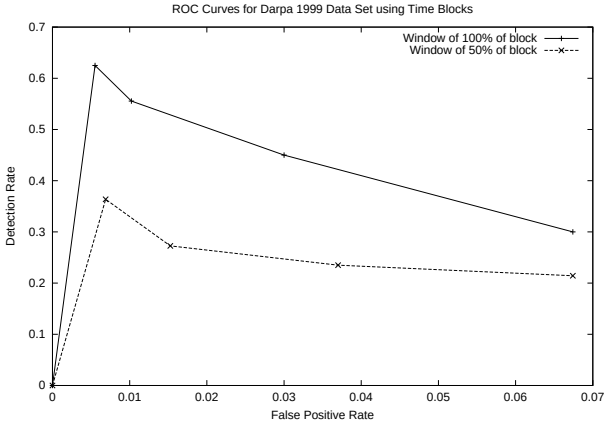

Fig. 4. Experiments using blocks of data based on different time periods.

VIII. DISCUSSION

In running the experiments, we used two types of blocks, time and amount. In block sizes base on time, we looked at 1, 2, and 10 minute blocks of traffic. Looking at the results in Figure 4, one notices that with rough rules, as information increases per block, the anomalous packets are lost in the background noise. Decreasing the amount of packets seen, increases prediction accuracy. In the second set of experiments, in blocks based on amount of records(figure 5) this is also true, although at a slower rate. Here, blocks were tested based on size of 100 to 500 packets. In addition, evaluating all the packets in the block increased accuracy, while looking at 50% of the block increased throughput at the expense of accuracy, as expected. Surprisingly, in blocks composed of n records, increasing the amount per block and decreasing the percentage evaluated increases performance. We feel that this is due to the nature of the traffic of the DARPA data, although further testing is needed.

ISBN 0-7803-9814-9/$10.00

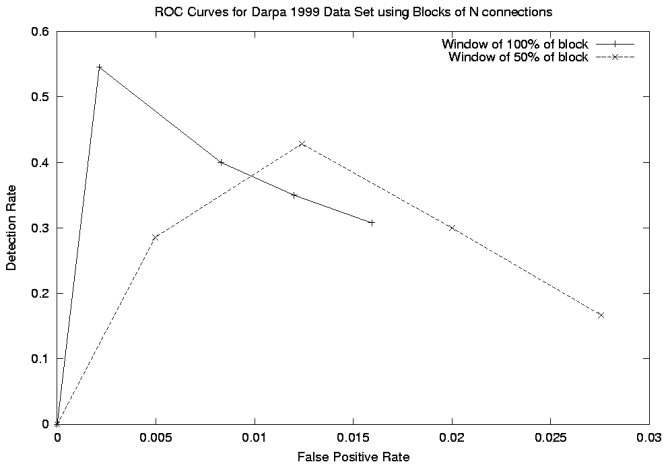

Fig. 5. Experiments using blocks of data based on number of records.

The contribution made by this paper is an architecture and a method to detect and alleviate denial of service attacks using signature-based methods on a time-partitioned block of data. This method has the advantage of being computationally inexpensive and having reasonable detection rate. The results in Table II depict an overall accuracy that is relatively low and this is mainly due to the coarse set of rules. We are confident that with a more extensive set of features and better model, accuracy will be improved.

IX. Future Work

We are in the process of implementing different learning algorithms (Naive Bayes, Neural Nets) on the data sets and will be reporting the results in subsequent reports. Specifically, we are looking at using a clustering[18], [19] method for detecting DOS attacks within the network data. We make two basic assumptions, mainly that each attack consists of a number of packets, which are similar to each other, and secondly, different than other packets over most of the pre-determined features. The set of features (such as number of connections per IP, etc.) is considered to be known in advance, but their particular values, as well as what constitutes malicious value of the feature, are considered unknown.

In our initial implementation, we have decided to concentrate on TCP attacks. For testing purposes, we consider attacks that involve normal values of each single packet but abnormal values of the statistical measures, such as number of SYNs, number of packets etc. Even though our system will eventually be used to detect unknown and new distributed attacks, we need to use known attacks, to estimate the efficiency of the method.

The overall network-wide view of the DFA leads to some interesting applications that are independent of its current realm of model generation. One of the most interesting possibilities, is to use the DFA as a load balancer in the network. We mention the possibility in the paper but have not carried out an implementation yet. This feature would allow the DFA to redirect heavy traffic flows to other servers. Another interesting concept is to have the DFA's randomly roving about the network. This prevents the generation of a single point of failure for this system as the DFA can randomly spawn in different locations. All of these additional tasks would increase the computational burden of the DFA and thus trade-offs would have to be made between the complexity of the model generation algorithms and the benefits of load balancing and system resilience to failure during an attack.

X. Acknowledgments

We would like to thank Eleazar Eskin for his effort in this research and to C.G.H. and Y. Hershkop in helping with the writing.

References

[1] D. Moore, G. M. Voelker, and S. Savage, "Inferring internet denail-of-service activity," in *Usenix*, 2001.
[2] "The cert coordination center."
[3] J. D. Howard, *An Analysis of Security Incedents on the Internet*. PhD thesis, Carnegie Mellon University, Aug 1998.
[4] D. Clark, "The design philosophy of the darpa internet protocols," in *Communication architecture and protocols*, pp. 106–114, 1988. Stanford, CA.
[5] G. C. Kessler, "Defenses against distributed denial of service attacks." SANS Institute.
[6] S. Gibson, "The strange tale of dos attacks against grc.com." http://grc.com/dos/grcdos.htm, 2001.
[7] O. Spatscheck and L. L. Peterson, "Defending against denail of service attacks in scout," in *Proceedings of 3rd USENIX/ACM*, pp. 59–72, 1999.
[8] P. Ferguson and D. Senie, "Defeating denial of service attacks which employ ip source address spoofing," 2000.
[9] C. Perkins, "Rfc 2002: Ip mobility support." http://www.faqs.org/rfcs/rfc2002.html, 1996.
[10] "Ddos enforcer," 2002. Mazu Networks.
[11] "Arbor networks." http://www.arbornetworks.com.
[12] W. Lee and S. Stolfo, "Data mining approaches for intrusion detection," in *Proceedings of the 7th USENIX Security Symposium*, (San Antonio, TX), 1998.
[13] I. K. Moskowitz, M. H. Kang, L. W. Chang, and G. E. Longdon, "Radomly roving agents for intrusion detection," tech. rep., NRL CHACS, 2001.
[14] V. Razmov, "Denial of service attacks and how to defend against them." Survey Project Paper, Dept. of Computer Science and Engineering, May 2000.
[15] M. G. Schultz, E. Eskin, and S. J. Stolfo, "Mef: Malicious email filter - a unix mail filter that detects malicious windows executables," in *FREENIX Track*, 2001.
[16] "Darpa intrustion detection sytem evaluation 1999."
[17] R. P. Lippmann, R. K. Cunningham, D. J. Fried, I. Graf, K. R. Kendall, S. W. Webster, and M. Zissman, "Results of the 1999 darpa off-line intrusion detection evaluation," in *Results of the 1999 darpa off-line intrusion detection evaluation*, (West Lafayette, IN), 1999.
[18] L. Portnoy, E. Eskin, and S. J. Stolfo, "Intrusion detection with unlabeled data using clustering," in *ACM Workshop*, 2001.
[19] C. Taylor and J. Alves-Foss, "Nate - network analysis of anomalous traffic events, a low-cost approach." NSPW, 2001.