3.2 Detection 5 h 57 Our ability to withstand the denial of service attack is greatly affected by our ability to correctly identify the presence of attack traffic and to filter it out. The whole process of detection and filtering is also greatly influenced by our choice	3.2 Phát hiện Khả năng chịu đựng tấn công từ chối dịch vụ chịu sự chi phối mạnh bởi khả năng của chúng ta trong việc xác định chính xác sự hiện diện của lưu lượng tấn công (giao thông mạng khi bị tấn công) và lọc nó. Toàn bộ quy trình phát hiện

for a monitoring point. Certainly, when we deploy any kind of detection mechanism at a target's network, it is relatively easy to identify attack traffic as it aggregates there. On the other hand, the sheer volume of that traffic makes filtering, allowing legitimate traffic and dropping or rate-limiting attack traffic, a very complicated and process-consuming matter, as usually we need to process every single packet.

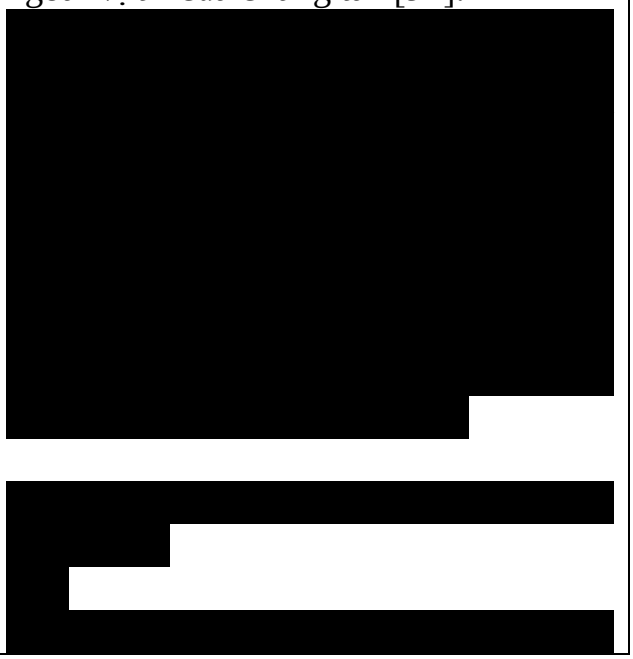
For example Paul Holbrook, Director for Internet Technologies for CNN said regarding the famous denial of service attack in February of 2000: "In our case, what caused us trouble was not that we weren't filtering it out. We were filtering it, but the problem was that the routers were so busy filtering that that in itself compromised us. The routers still have to process each packet. The cure was putting the filter on a bigger router outside of our site" [32].

As shown in Figure 4 [27], we can put our monitoring point at four different locations. The first, most obvious one, right at the target's network, provides higher degree of confidentiality in determining denial of service activity, while effectiveness of packet filtering is rather low as we have to cope with large volumes of normal and attack traffic throne together.

Figure 4. Possible locations for performing detection and filtering [27]

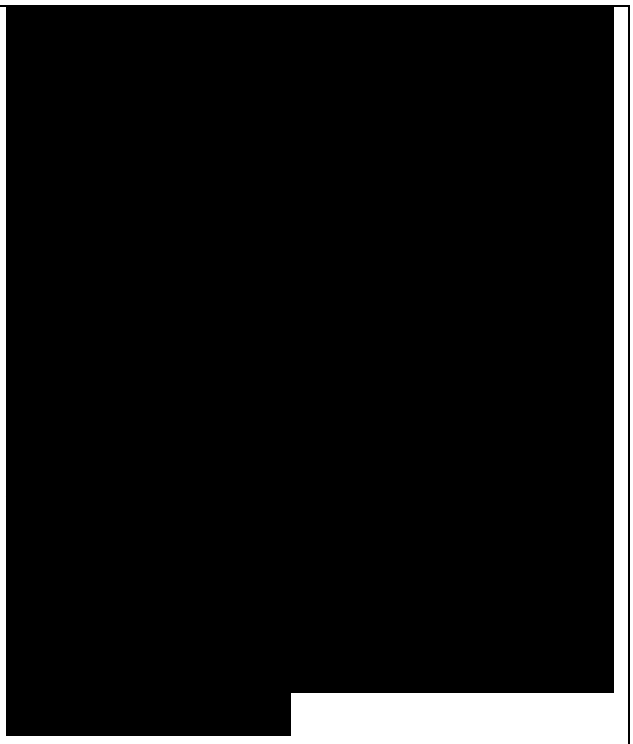
On the opposite end is the location at

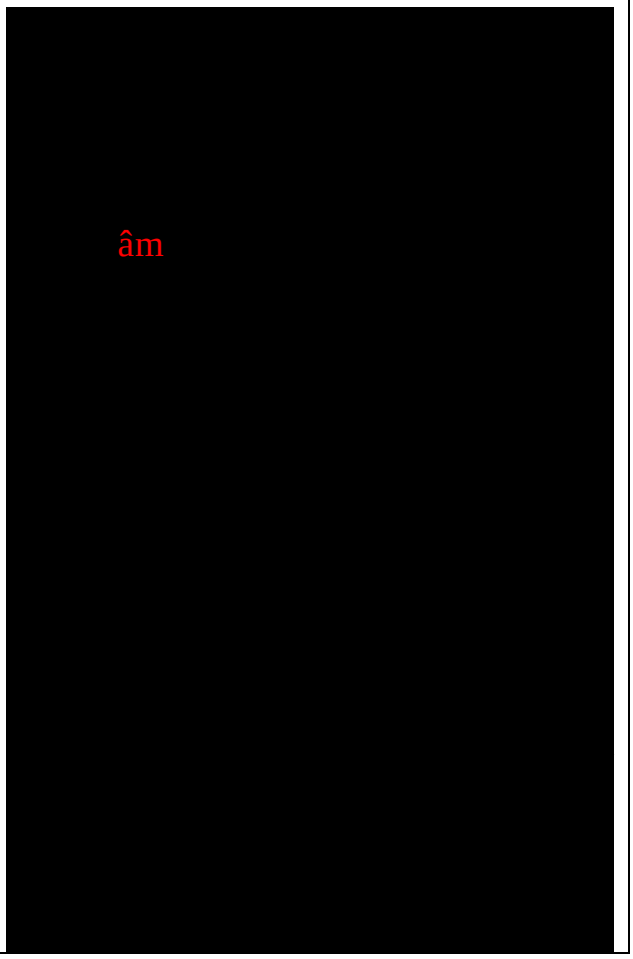
và lọc cũng chịu tác động mạnh bởi sự chọn lọc điểm giám sát của chúng ta. Tất nhiên, khi chúng ta triển khai bất kỳ loại cơ chế phát hiện nào tại mạng của mục tiêu, việc xác định lưu lượng tấn công tương đối dễ dàng vì nó tập hợp ở đó. Mặt khác, lưu lượng lớn đó làm cho quá trình lọc, cho phép lưu lượng hợp lệ và bỏ qua hoặc giới hạn tốc độ lưu lượng tấn công trở thành vấn đề rất phức tạp và đòi hỏi tốn nhiều công sức xử lý, vì thông thường chúng ta cần xử lý từng gói tin. Ví dụ Paul Holbrook, Giám đốc Công Nghệ Internet cho CNN đã nói về một vụ tấn công từ chối dịch vụ nổi tiếng vào tháng Hai năm 2000: "Trong trường hợp của chúng tôi, điều gây rắc rối không phải là chúng tôi không lọc nó. Chúng tôi đang lọc nó nhưng vấn đề là các bộ định tuyến quá bận lọc đến nỗi chính điều đó lại phản tác dụng. Các bộ định tuyến vẫn phải xử lý mỗi gói tin. Cách xử lý là đặt một bộ lọc trên bộ định tuyến lớn hơn bên ngoài vị trí của chúng ta" [32].



attack source networks, where attack flows originate. Unless there are a relatively large number of nodes participating in the attack, we have a small chance of detecting any attack activity, but if we do manage to detect attack flows, then it is more effective to filter them out at the source. The effectiveness of packet filtering declines rapidly on the packet route to destination, because more normal packets would also be dropped, as no filtering technique is 100% accurate. Two other locations - target's ISP network and further upstream networks - represent compromises/tradeoffs between effectiveness of packet filtering and effectiveness of attack detection.

Generally the effectiveness of any detection mechanism can be rated by such parameters as detection time, false positive ratio (FPR) and false negative ratio (FNR). The false positive ratio is the number of packets classified as attack packets (positive) by our detection engine that our found to be normal (negative), divided by the total number of confirmed normal packets. The false positive ratio is the number of packets classified as normal (negative) by detection engine that are confirmed to be attack packets (positive), divided by the total number of confirmed attack packets. For the effective detection mechanism these ratios should be as low as possible. For the measure of filtering effectiveness there is the normal packet survival ratio (NPSR) which is a percentage of normal packets that still get through to the target while

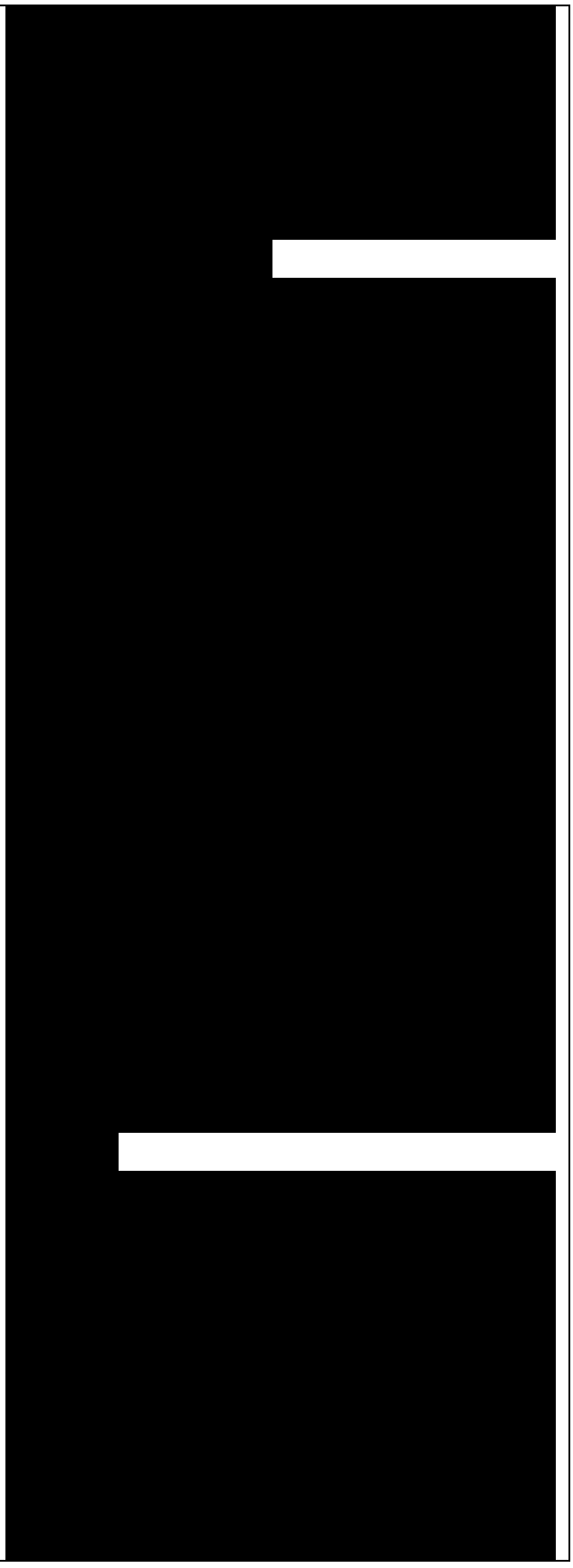




that target is under denial of service attack. Higher NPSR values correspond, obviously, to higher ratios of normal traffic unaffected by the attack [27].

Designing detection mechanisms it is necessary to implement not only the ability to distinguish between normal and attack traffic, but also the ability to make distinction between attack traffic and traffic associated with flash events. Flash event is defined as “a large surge in traffic to a particular Web site causing a dramatic increase in server load and putting severe strain on the network links leading to the server, which results in considerable increase in packet loss and congestion” [11]. A typical example of a flash event would be a rush-in of users to the major news site, such as CNN or BBC in the case of major terrorist attack, like September 11. In this case we would expect those sites to be swamped with numerous requests.

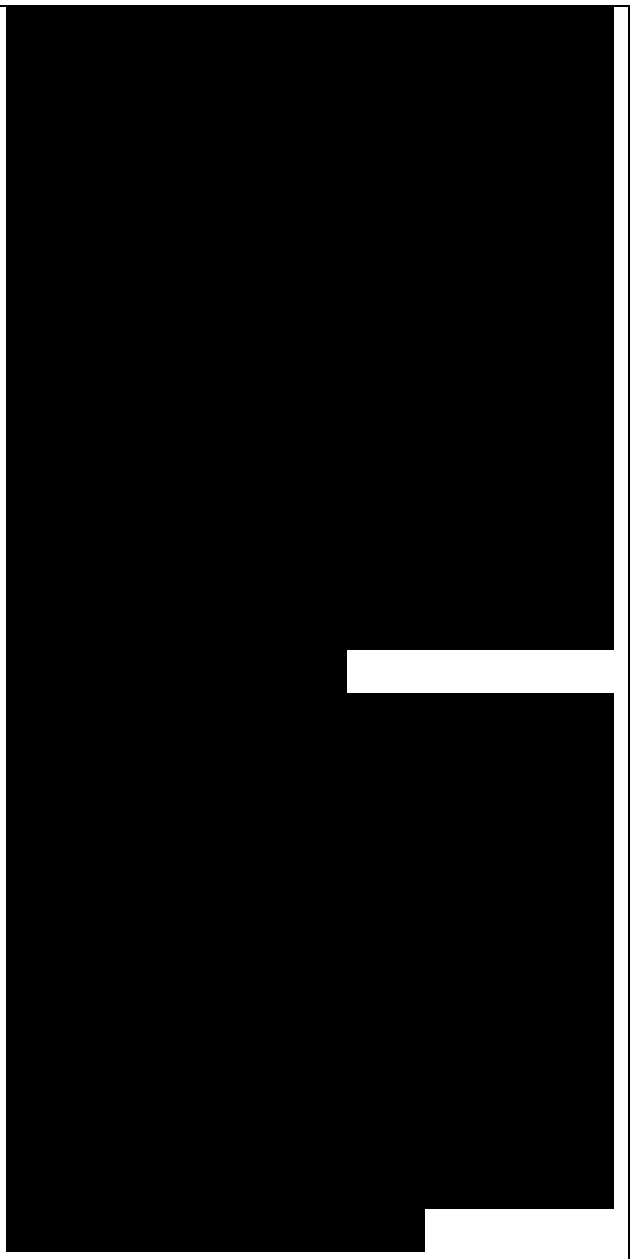
The key difference between a flash event and denial of service attack is that in the former case the requests are legitimate but this observation does not help unless we are aware of some other distinctions in behavior of both events. In [11] an analysis of a number of network traces was done which revealed quite interesting conclusion, namely that a Web site receives many more requests



from previously seen clusters, i.e. subnets/local networks during a flash event than during an attack. Important thing here is that we compare number of clusters and not the number of individual clients, as the percentage of individual clients is small in all events. From the common sense it seems that legitimate requests are from those clusters where at least some of the users have already visited the Web site while in the case of denial of service attacks addresses are random - as the attacker was able to install agents at some randomly chosen hosts.

As we are interested particularly in detection of denial of service attacks, let us consider some of the existing solutions. Usually detection techniques can be put into one of two distinct groups: DDoS attack-specific detection which obviously is based on the some signatures exhibited by attack flows and anomaly-based detection which, as the name suggests, observes normal traffic and monitors it for any abnormalities which would naturally be a sign of an attack.

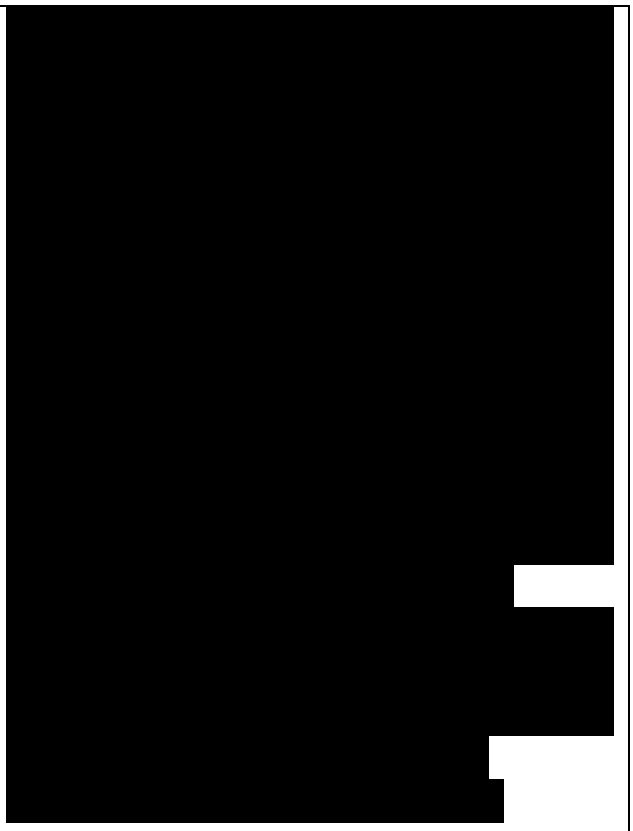
The attack-specific detection relies on a number of signatures (chữ ký, dạng tấn công) to be found in the passing traffic - detection engine is constantly trying to match observed traffic to a signatures database. The main advantage of such approach is that the known attacks (for which we have signatures) are easily and reliably identifiable. On the other



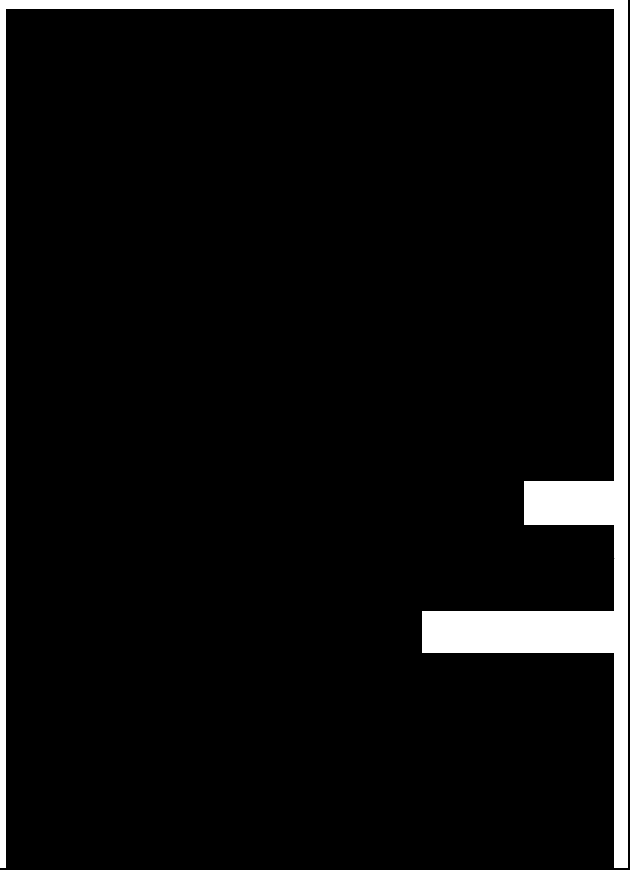
hand, to be effective the database has to be kept up-to-date and it is not possible to recognize new attacks. There are currently a number of popular solutions for monitoring traffic which use signatures to detect denial of service attacks. As detection is based on signatures, those solutions are typically a part of much broader security suits. Two of the examples are proprietary IBM Proventia Network Intrusion Prevention System (IPS) [33] and open source Intrusion Detection System (IDS) Snort [34].

Anomaly-detection systems can be further divided into separate categories, such as statistical analysis based and rate limiting based.

Statistical analysis based:



- NOMAD[35] is one of the oldest mechanisms which relies on statistical analysis of IP packet header information, such as time-to-live (TTL), source/destination address and packet length as also as router's timestamps. Detection algorithms use path changes, flows shift and packet delay variance to detect traffic overload, router misconfiguration/malfunction and component failure.

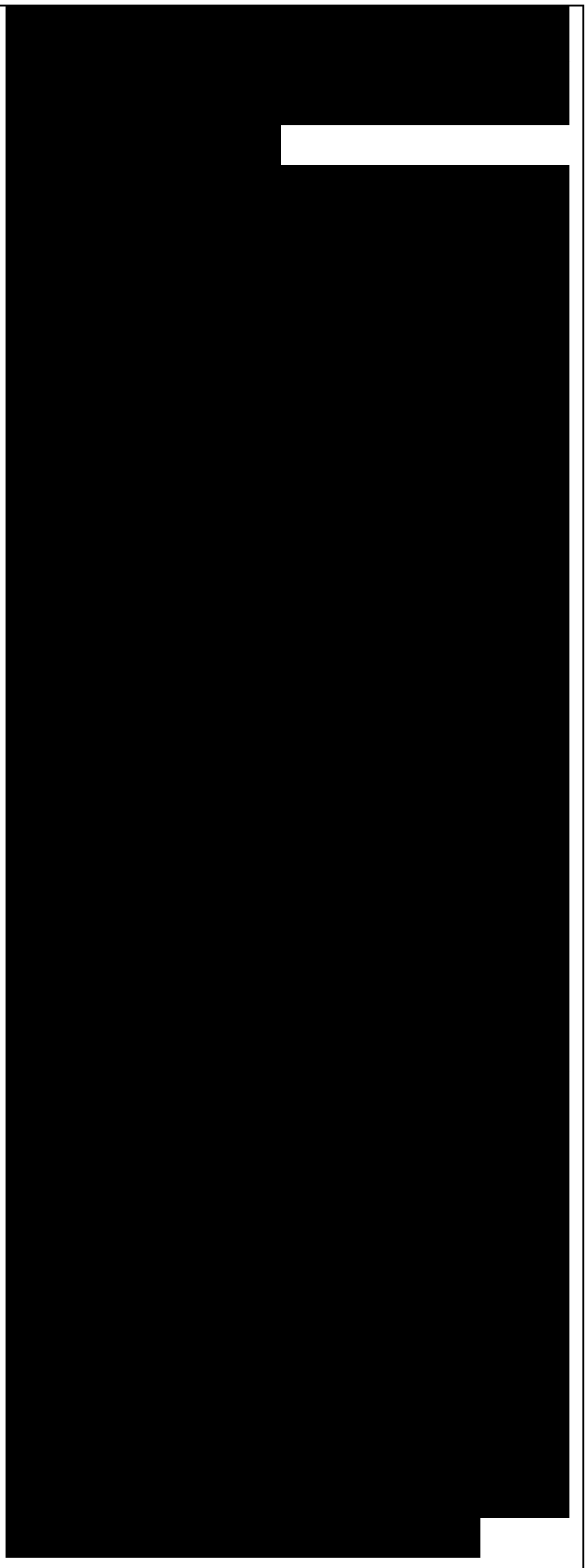


As this paper was published in 1999, there is no explicit mention in it of denial of service attacks.

- A correlation between traffic patterns at the target and traffic patterns at the source is a basis for another approach [36]. This scheme uses time series analysis on the data collected

from routers into Management Information Base (MIB) to detect statistical patterns.

• Sequential change-point detection. Algorithms of this group use the change in statistic during the denial of service attack as an indication of attack happening. The theoretical foundations are presented in [37] and [5]. The idea behind this approach is that any rapid change in the traffic during an attack results in the corresponding change of statistical models, thus a deviation from "normal model" is a sign of an attack. We collect any related statistic at our observation point during fixed time periods. The objective of sequential change-point detection is to decide if observed time series remain statistically consistent and if not, to determine time moment at which change occurred. As the object of research is Internet traffic, for which we do not have simple parametric model, as Internet traffic is of complex stochastic nature, instead we rely on the non-parametric CUSUM (Cumulative Sum) method for an attack detection [5]. A description of CUSUM algorithm is presented in section 4.1. Detection schemes, which use CUSUM algorithm for DDoS attack detection, employ as an observation statistic either number of new IP addresses at the observation point or in the case of detection SYN flooding attacks the number of SYN-FIN pairs [9, 10].

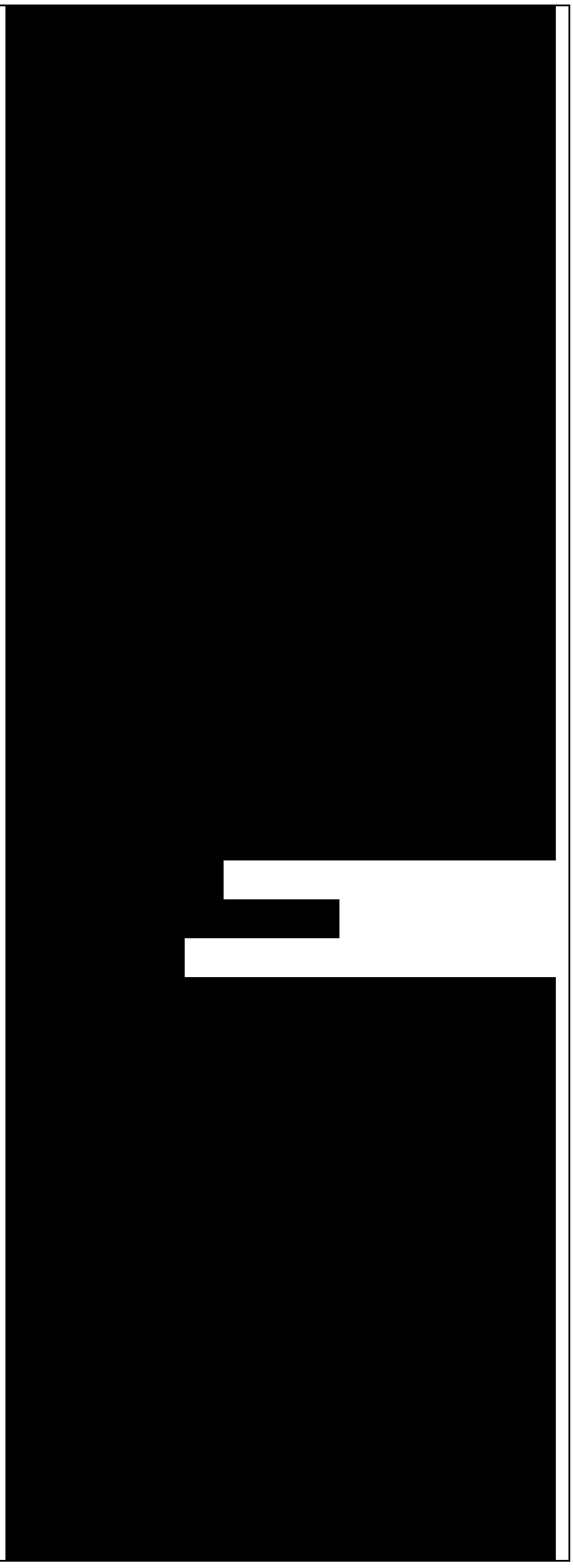


Rate limiting based:

- One of the representatives of this group is D-WARD - DDoS Network Attack Recognition and Defense. [38] This approach is based on the assumption that denial of service attacks have to be stopped as close to the source as possible. D- WARD is installed at the source router in the deployment network. The observation component of the detection engine monitors all packets passing through the router and collects statistics on two-way communications between sets of local addresses and the rest of the Internet. Periodically statistics are compared to the legitimate traffic models and the decision is made whether flows and connections are legitimate or not. The observation component passes the information to the rate-limiting component, which in turn makes a decision to set, modify or remove the rate limit on the flows. Rate-limit rules are passed on to the traffic-policing component, which uses a list of classified-legitimate connections to enforce the rules on the allegedly attack traffic, trimming the outgoing flows.

- Another example of detection technique based on rate limiting is MULTOPS [39]. MULTOPS uses the assumption that the rates of incoming and outgoing connections under normal conditions should be in balance. Under this scheme network devices maintain a multilevel tree, keeping track of certain network statistics and store results in nodes corresponding to subnet prefixes

at different aggregation levels. As soon as the attack happens, there is a disproportionate difference in rates of incoming and outgoing traffic for certain flows and as a response those flows are rate limited. As opposed to D-WARD, MULTOPS can be implemented either at source side or target side. Though by design D-WARD and MULTOPS are quite similar, there are some differences. For example, MULTOPS imposes fixed, non-selective limit on incoming or outgoing traffic as opposed to D- WARD which uses selective and dynamic response. Unfortunately, because of the use of a disparity between in and out flows as a key parameter, both approaches cannot prevent proportional attacks, i.e. attacks with balanced rates of incoming and outgoing connections.



4.1 CUSUM algorithm

CUSUM algorithm in our deployment uses a number of new source network addresses as its statistic. Before we continue with discussion of algorithm's operation in ns-2 network simulator - actual source code implementation, let us consider the placement of the detection monitoring point. We can position our detection engine either at the first-mile or last-mile leaf routers, which are, basically, the gateways to and from the Internet for the local intranet. Every leaf router can serve in both capacities, depending on the direction of incoming and outgoing

traffic. For example, for the traffic leaving the intranet the leaf router is the first-mile router. Otherwise, for the incoming traffic into the intranet the leaf router is the last-mile router. Usually any detection engine is placed on the link connecting the Internet to the local intranet, monitoring the bidirectional traffic.

In ns-2 simulations we are running we use the last-mile router scenario, in which we intercept all incoming traffic. We examine all packets coming to the leaf node, stripping the header and retrieving the source address of any intercepted packet. This operation is done constantly, so we have at any moment an up-to-date database of all observed source network addresses. At fixed time observation intervals A we check this database, comparing it with the database of source addresses, which was recorded at the previous time. From a number of consecutive check-ups we as a result have a random sequence representing the number of new IP addresses in a time interval A. To smooth the resulting time series in regard to traffic variations, we normalize those values by dividing them by the overall number of discovered addresses at any given time. Resulting sequence xn under normal conditions, when the number of newly discovered addresses is relatively small, will have mean value a which is quite close to 0 (as indicated in Figure 5) [6]. As soon as there is an influx of new addresses during an attack, the mean value will experience the step-like minimum

increase by value h so the mean value would be now a + h.

As one of the assumptions of CUSUM algorithm is that mean value of a random sequence is negative during normal operation, and becomes positive during change, we transform our random sequence xn into new random sequence zn (Figure 6) [6]. This transformation is zn= xn - P, where P is such a constant that a = a - P < 0.

Figure 6. Increase in mean value of sequence zn in the presence of an attack [6]

CUSUM algorithm measures the values of the statistic used (in our case it is the number of new addresses) which overlaps the threshold N for the purpose of attack detection, using recursive expression (see Figure 7) [6].

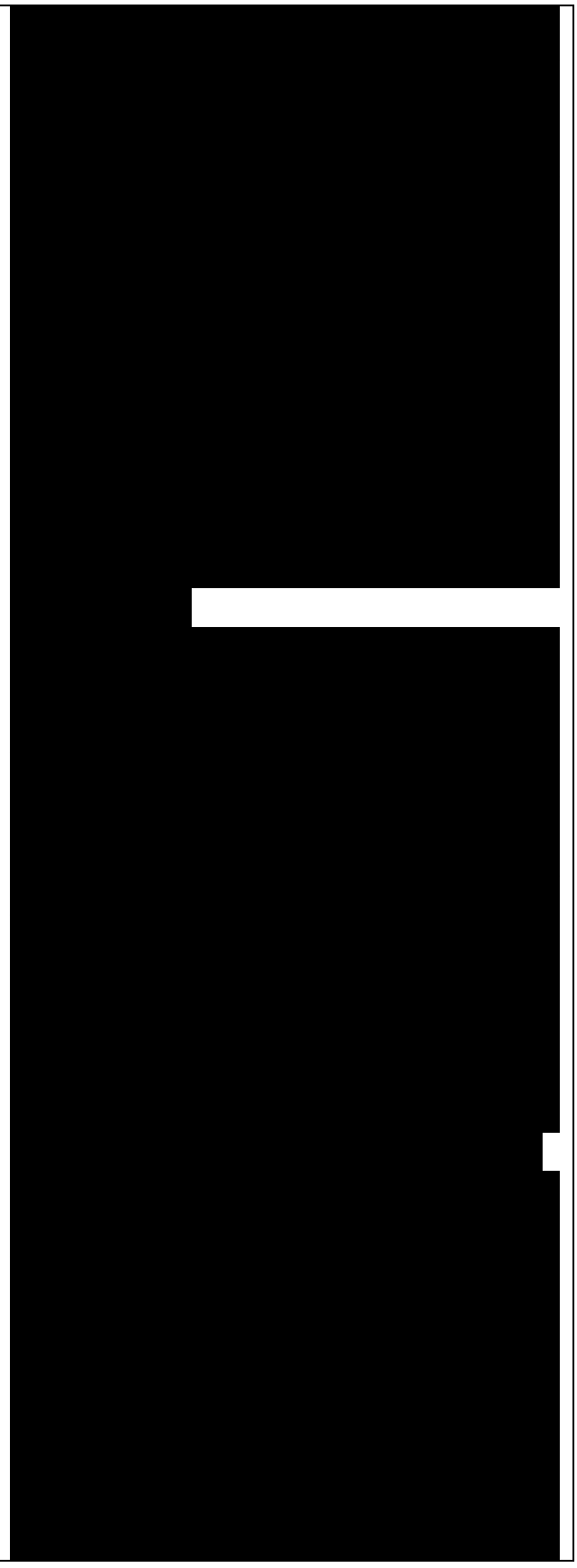
If at some time we have a large value of yn, exceeding threshold N then it is a good indication that indeed our system is under distributed denial of service attack. Suppose, as presented in Figure 7, that the attack begins at time m. Then the value of yn will increase until it reaches the threshold at time tn - a moment when attack is declared and it is time to implement some defensive measures. As can be seen, the operation of this algorithm is rather straightforward and very simple.

As an example of the dependence of sensitivity of our algorithm to the observation time interval there are results of network simulations for the case where the parameters used were: P

= 0.1 and N = 0.1 for different observation periods (5 and 2 seconds). It is necessary to note, though, that the use of particular values for the threshold N and the offset P here and in the simulation runs results of which discussed later does not justify the blind use of those values for the CUSUM algorithm for any general case. The reason is that we lack a parametric model for xn and, correspondingly, for zn. As such any discussion of optimal values for N and P makes sense only if we consider the onsite conditions, i.e. take into consideration the local network conditions.

The graph of yn was drawn for the same traffic generators and same topology in both cases. The burst nature of the denial of service attack is modeled arbitrarily by directing first wave of the attack in the interval from 50.3 till 61.2 seconds and the second wave from 72.3 till 80.2 seconds of simulation scenario. The length of both attacks in the range of about 10 seconds was chosen to allow our algorithm to detect the change of measured statistic for the case of observation time of 5 seconds and less.

When observation time interval was 5 seconds (Figure 10), the time delay between the start of the attack at 50.3 seconds and the moment when yn reaches the threshold value (at 54th second) is 3.7 seconds - detection time. For the case when observation interval was 2 seconds, the detection time is 1.7 seconds (52 - 50.3 seconds). Also it can be noted that when the observation

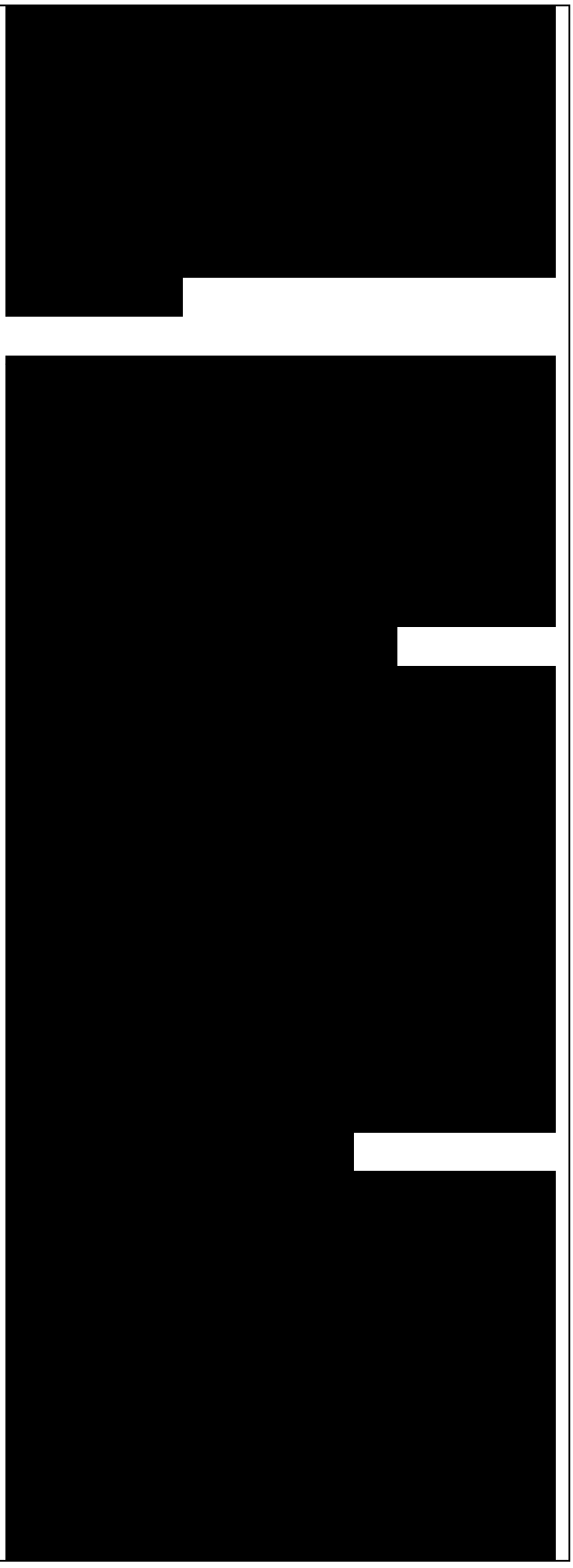


period is 5 seconds, the attack flag turns "up" at 54th second and is maintained in this state till the end of the simulation, so the detection engine reports the attack when in reality it has already stopped. On the other hand, smaller observation time period of 2 seconds (Figure 11) resolves the presence of the second wave of the attack.

As we can see, when we have smaller observation time interval, we are able to distinguish finer structure of denial of service traffic, especially its pulse or burst nature. Bigger time intervals seem to mask this attack feature.

Figure 12 and Figure 13 present graphs yn for the case P = 0.05 and N = 0.1 for same different time intervals of 5 and 2 seconds (network topology and attack flows are preserved). Comparing pairs of graphs with different values of P we can observe that as expected for the smaller P the peaks of yn have bigger values. The less is the offset of zn into negatives, then consequently more likely is the value of yn to exceed zero and be counted towards larger value to indicate an attack.

The same kind of observations for the same kind of topology and attack flows was done for the case of P = 0.05 and P = 0.1 for N = 0.2. Results are presented in Figures 14 - 17. As expected, if we raise the detection threshold, the detection time increases significantly. Detection delay is now about 24 seconds for both cases (for observation time intervals of 5 and 2 seconds)



mainly because of the second attack wave. It seems that the first wave was not sufficient enough to raise the value of yn above the detection threshold.
Another pair of observation of behavior yn is done for different values of P =0. 1 and P = 0.05 for N = 0.3. Results are shown in Figures 18 - 21. Evidently, the threshold value of 0.3 was set too high and our detection engine was not able to detect any attack at all regardless of P, though as shown in Figure 18 the second peak of yn almost reached the detection threshold.

One important note to make is that in our CUSUM algorithm implementation we artificially trim the cumulative sum of the sample statistic in the case when during an observation period there are no new network source addresses and the attack flag is in "up" position. The reasoning behind that decision is that we presume that if there are no new addresses recorded during time Δ, there are two possibilities for that. First off, the attack really stopped. As soon as it happens we need to set the attack flag in "down" position, essentially notifying the filtering engine to stop its activity and lessening the CPU load. So if there are no new addresses we assign the value of the threshold to the variable yn – our statistic but the attack flag is still in "up" position – just a precaution, really. If during next time interval there is still no new addresses discovered or their number is substantially low enough to merit statistic's value to drop below threshold, then we declare the

attack flag “down” – attack has stopped for real. Second, if the absence of new addresses during time Δ was no more than a fluke, and the attack is still in progress, then at next observation time the statistic value yn will be easily upset over threshold and the attack flag still going to be kept “up”.

This tweak of the CUSUM algorithm was done to address the feature of the conventional algorithm, briefly discussed in [6] and evident from the recursive expression for yn, namely that the conventional scheme can not accurately detect the end of the attack because the CUSUM algorithm decreases very slowly. As an illustration consider **Figure 22**, where two graphs for yn are plotted together: one for the conventional CUSUM algorithm and another one for our tweaked version – in both cases the simulation setup is the same, i.e. first wave of the attack is in the interval from 50.3 till 61.2 seconds and the second wave from 72.3 till 80.2 seconds of simulation scenario. We see that our tweaked CUSUM algorithm identifies the end of the attack more accurately.

To summarize, it can be noted that the choice of β influences our detection sensitivity – the larger β we have (the bigger offset in negatives of a sequence *zn*) – the less likely a value will appear in *zn*that is going to be more than zero. In its turn, it is less likely that the test

statistic will accumulate enough to merit the attack flag.being set “up”. The larger the threshold N, the more time delay before detecting an attack. Set threshold too high, and an attack with sufficiently low intensity will not be detected at all.	[redacted]