On the Effectiveness of DDoS Attacks on Statistical Filtering

Qiming Li
Temasek Laboratories
National University of Singapore
liqm@comp.nus.edu.sg

Ee-Chien Chang
Department of Computer Science
National University of Singapore
changec@comp.nus.edu.sg

Mun Choon Chan
Department of Computer Science
National University of Singapore
chanmc@comp.nus.edu.sg

***Abstract*— Distributed Denial of Service (DDoS) attacks pose a serious threat to service availability of the victim network by severely degrading its performance. Recently, there has been significant interest in the use of statistical-based filtering to defend against and mitigate the effect of DDoS attacks. Under this approach, packet statistics are monitored to classify normal and abnormal behaviour. Under attack, packets that are classified as abnormal are dropped by the filter that guards the victim network. We study the effectiveness of DDoS attacks on such statistical-based filtering in a general context where the attackers are "smart". We first give an optimal policy for the filter when the statistical behaviours of both the attackers and the filter are static. We next consider cases where both the attacker and the filter can dynamically change their behaviour, possibly depending on the perceived behaviour of the other party. We observe that while an adaptive filter can effectively defend against a static attacker, the filter can perform much worse if the attacker is more dynamic than perceived.**

I. Introduction

A Distributed Denial of Service (DDoS) attack is an attack where many compromised hosts send large amount of traffic to the victim network elements such that the resources of the elements are exhausted and the performance seen by legitimate packets are severely degraded. Such attacks have been witnessed in the Internet, such as the recent attacks caused by the MyDoom virus (http://www.us-cert.gov/cas/techalerts/TA04-028A.html).

Statistical approaches to defend against DDoS attacks have been proposed ([1], [2], [3], [4], [5]). In statistical approaches to defend against DDoS attacks, the statistics of packet attributes in the headers, such as IP address, time-to-live (TTL), protocol type etc., are measured and the packets deemed most likely to be attack packets based on these measurements are dropped. Such approaches often assume that there are some traffic characteristics that are inherently stable during normal network operations ([2],[5]). Therefore, during a DDoS attack, "abnormal" traffic can be detected based on the assumption of stable traffic characteristics, and the packets that are most likely to be illegitimate are dropped by the filter that guards the victim network.

It is unlikely to have a fully automated mechanism that can successfully defend against all DDoS attacks. When an attack is detected, human intervention is eventually necessary to protect the victim network. Most of the existing automated defence mechanisms focus on the few hours after the attacks commence, before a human expert is able to respond.

In this paper, we first study the effectiveness of DDoS attacks on statistical-based filtering in a general context where the filter uses an optimal dropping policy with respect to static attackers. We generalize the problem to the cases where both the attackers and the filter can dynamically change their behaviours, possibly depending on the perceived behaviour of the other side.

Specifically, the following cases are considered, in the order of increasing sophistication of the attack. We call the traffic characteristics assumed by the filter under normal network operations the *nominal traffic profile*.

- **Static Attackers and Static Filter**. We consider scenarios where the attackers and the filter decide on the specific attack traffic distributions and the filter policy to use respectively, and do not change their decisions.
- **Static Attackers and Adaptive Filter**. Similar to the previous case, but the filter can learn the attack traffic distributions and adapt its policy accordingly.
- **Dynamic Attackers and Adaptive Filter with No Feedback**. Similar to the previous case except that the attackers are allowed to change the attack traffic distributions.
- **Adaptive Attackers and Adaptive Filter with Feedback**. Attackers are allowed to change their attack traffic and also know if the attack packets are dropped or allowed to pass through.

We show that the success of a static filter is highly dependent on its capability to estimate the static attack distributions. We further show that while an adaptive filter can effectively defend against static attackers, it can perform much worse than a static filter if the attackers dynamically change the distributions of the attack packets to trick the filter into setting the wrong policies.

The success of these attacks relies on the assumption that even with minimum knowledge of the filter's nominal traffic profile, the false accept rate is not insignificant unless the false reject rate is made arbitrarily high. We demonstrate the validity of this assumption using some packet traces collected from different locations.

This paper is organized as follows. In Section II, we describe related works in detecting and defending against

DDoS attacks. In Section III, we present a generic model for statistical-based filtering. In Section IV, we present the optimal filter policy for static attackers and filter. In Section V, we study whether adaptation improves the filter performance. In Section VI, we present attack strategies for attackers who can probe the filter using feedback. In Section VII, evaluations of different attack strategies and filter policies are presented. Finally, we conclude in Section VIII.

II. Related Work

DDoS attacks are a well known problem and a good overview of common DDoS attacks can be found in [6].

Defending against DDoS attacks often involves detection and response. There are a number of statistical approaches for detection of DDoS attacks, including the use of MIB traffic variables [7], IP addresses [4] (which assumes that attack traffic uses *randomly* spoofed source addresses), IP addresses and TTL values [3] and TCP SYN/FIN packets for detecting SYN flood attacks [8]. More general approaches towards statistical detection and response can be found in [2] and [5] where statistics in packet (network and transport layers) attributes can be used for both detection and setting of filtering policy for packet dropping. In [2], entropy and Chi-Square statistics are used to differentiate between attack and normal packets while [5] computes the *conditional legitimate probability* of a packet (the likelihood that a packet is legitimate given a baseline nominal traffic pattern). The D-WARD approach [1] uses, in addition to network and transport header statistics, application layer knowledge to implement the filter policy.

Another way to defend against DDoS attacks is the use of pushback. The idea is that if the source of the attacks can be identified and traceback incrementally hop-by-hop to the source (or as close as possible), then rate limiting can be used to limit the scope and damage of the attacks. [9] proposes the concept of high bandwidth aggregates for such identification. [10] proposes an IP traceback scheme where packets are randomly marked for tracking the routes of the attack packets. When sufficient packets are marked, this approach allows a victim to identify the network path(s) traversed by the attack traffic without requiring operational support from ISPs. Finally, [11] describes a route-based distributed packet filtering (DPF) scheme for implementing pushback.

III. Formulation of Attacks on Statistical-Based Filtering

In this section, we give a model for the game between a filter and the attackers. Our analysis centers around the notion of (1) $\widetilde{A}$, the attack packet distribution *perceived* by the filter, (2) A, the actual attack distribution, and (3) D, the policy employed by the filter, based on which packets are dropped. The game between the attackers and the filter is essentially a game of deception and guessing of $\widetilde{A}$, A and D, either in a static or a dynamic setting.

Fig. 1 shows the DDoS model assumed. The filter, residing between the network and the victim, sees the mixture of the actual attack distribution A and the legitimate traffic Q. Based

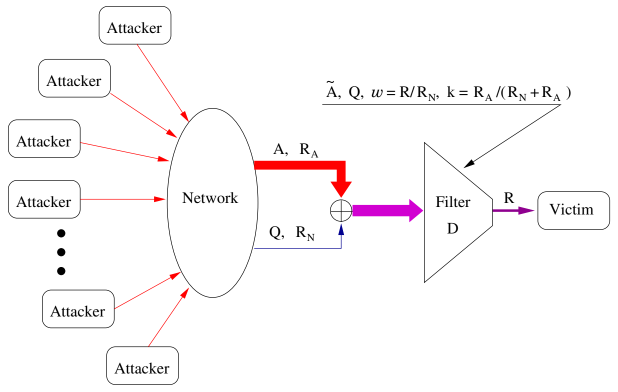

Fig. 1. DDoS attacks on statistical filter

on an estimation $\widetilde{A}$ of the attack distribution, the filter devises a policy D to decide if a packet should be dropped or allowed through. R is the link bandwidth between the filter and the victim. It is necessary to limit the rate of the packets allowed through to R, since the victim may not be able to cope with a higher rate.

A. Filter's Objective

A filter takes in a packet and decides whether to accept or reject (drop) it. If a packet is accepted, it can pass through the filter and reach the victim. There are a few issues regarding the notion of a good filter. Firstly we define *false accept* α to be the probability that an attack packet is accepted, and *false reject* β to be the probability that a legitimate packet is rejected. The objective of the filter is to keep both α and β as small as possible, while maintaining overall accepted traffic rate to be within a predefined parameter R. However, these goals are conflicting, and different scenarios will place different level of emphasis on these goals. Under our model, the filter attempts to achieve a particular rate R, while minimizing the *effectiveness* e of the attack, which is a function of α and β, and possibly other parameters.

B. Filter Policy

Given a packet, the filter makes its decision based on the feature of the packet, which usually contains network and transport layer header attributes such as TTL, IP address (or prefix), packet size, port numbers, etc.. We refer to the algorithm that the filter uses to make its decisions the *policy*.

Due to the constraint in computing power, the filter is unable to adapt its policy rapidly to the changing traffic. Thus, we assume that the policy is fixed for at least a short period of time, for example, 5 minutes. We call such a short period of time wherein the policy is fixed a *slot*.

A filter policy can be carried out in the following way. For a given packet, its feature s is first extracted. Next, from a lookup table, the filter decides whether to accept or reject the packet. In other words, the policy is simply a function from U, the features space, to the decision $\{$accept, reject$\}$. We call such a policy *deterministic*. More generally, a *probabilistic* policy can be represented as a function D from U to the interval $[0,1]$. Given a packet with feature s, the filter computes the function $D(s)$ and accepts it with probability $D(s)$. A deterministic policy is a special case of a probabilistic policy when the value of $D(s)$ is 0 or 1.

We assume that the decision of whether a packet is dropped or accepted is independent on the outcomes of the previously received packets. Hence, the policy is *memoryless*.

However, in order to force rate limiting, the filter can not be completely memoryless. The total number of packets received and accepted so far can be monitored and used in adjusting the policy so as to achieve the desired rate. For example, the filter can employ the current memoryless policy for a short period of time and observe the number of packets received and accepted. If the rate of accepted packets is too high or too low, the policy is adjusted accordingly. The process is repeated until the desired rate is achieved. In this paper, we assume that the filter can adjust the policy and obtain the desired rate in 0 time. Nevertheless, due to the constraint in computing resources, the adjusted policy has to be efficiently obtained. We consider a combination of these two adjustments. The first adjustment simply adjusts the policy so that $D(s)$ for some feature s becomes zero. In the second adjustment, the filter policy is adjusted by multiplying $D(\cdot)$ by a constant.

C. Nominal Traffic Profile & Perceived Attack Distributions

In order to derive the policy, the filter also requires information about the traffic characteristics under normal network operations, or the nominal traffic profile.

Let Q, the *nominal traffic profile*, be the probability density function (p.d.f) of the nominal traffic. Hence, if s is the feature of a packet, then $Q(s)$ is the probability that such feature appears in the nominal traffic.

Based on $\widetilde{A}$ and Q, the filter can derive an optimal policy D (Section IV). Certainly, the optimality is based on the assumption that the perceived $\widetilde{A}$ is the same as the actual A, and the attack distributions remain unchanged within the slot.

D. Adaptive Policies and Dynamic Attack Strategies

The attackers generate packets and attempt to get as many packets to pass through the filter as possible. Since the filter employs a memoryless policy, it is sufficient for the attackers to find a distribution A, and generate packets according to A.

As the perceived $\widetilde{A}$ might not be the actual A, the filter may attempt to learn A and adapt its policy. On the other hand, the attackers may also employ a time-varying strategy, especially if they are equipped with the probing ability. As both sides can change their behaviours, based on the observation of the traffic or the assumption on the nominal traffic distribution, the main question is, does adaptation help the filter? In section V and VII-C, we argue that, filter adaptation is potentially more damaging, in the sense that the performance of the filter could be worse than without adaptation.

Due to limitations of computing resources, we assume that adaptation of policies is only performed at the beginning of a slot (Section III-B). Let $D_i, \widetilde{A}_i, A_i$ be the dropping policy, attack distribution perceived by the filter, and the actual attack distribution within the i-th slot respectively. During the transition from the $(i-1)$-th to the i-th slot, the filter may have access to the network traces in some or all the previous slots to derive the new policy D_i. We define the *learning window size* W to be the number of previous slots the filter has access to. At time 0, it is assumed that $\widetilde{A}_1$ is the uniform distribution.

E. Notations

Here is a summary of the notations used.

U:	Sample space. The set of all possible packet features.
s:	The feature of a packet, that is, $s \in U$. A feature could be a tuple of header attributes, e.g., packet length, TTL, source IP address, etc..
Q:	P.d.f of the nominal traffic distribution. $Q(s)$ is the probability that a legitimate packet has the feature s.
A:	Actual attack distribution, from which the attack packets are generated.
$\widetilde{A}$:	Attack distribution perceived by the filter. The filter derives its policy based on Q and $\widetilde{A}$.
D:	Policy employed by the filter in one slot. A slot is a short time period within which the policy remain unchanged. $D(s)$ is the probability that a packet with feature s is accepted in that slot.
α:	False accept. The probability that an attack packet is accepted.
β:	False reject. The probability that a legitimate packet is dropped.
e:	Attack effectiveness, which is $(\alpha+\beta)$ in this paper.
R_N:	Traffic rate of legitimate packets.
R_A:	Traffic rate of attack packets.
R:	Traffic rate of packets that the filter allows through.
k:	Probability that a received packet is an attack packet. That is, $k = R_A/(R_N + R_A)$.
ω:	Desired traffic ratio. It is defined as $\omega = R/R_N$.
W:	Size of the learning window. Number of slots of network traces that the filter has access to.

IV. Static Attackers and Filter

A. Optimal Policy

We first consider the case where the attackers decide on a static attack distribution and the filter employs a fixed policy. Both sides do not change throughout the duration of the attack. In order to derive the optimal policy in this case, we assume that the filter knows the nominal traffic profile Q and the perceived attack distribution is same as the actual attack distribution, i.e. $\widetilde{A} = A$. The filter also knows k, the proportion of attack packets in the received packets. The filter wants to maintain the overall accepted traffic rate such that the ratio of number of packets accepted over the number of legitimate packets is the desired traffic ratio ω. At the same time, the filter wants to minimize the *effectiveness* of the attack, which is a function of the false accept α, false reject β and possibly other weight parameters like k. A typical effectiveness is

$$e = (\alpha + \beta). \tag{1}$$

In the case where the attack distribution A is the same as the nominal traffic profile Q, we have $e = 1$. In the case where the filter randomly drops every packet with a fixed probability, we also have $e = 1$.

The optimization problem can thus be formulated as

$$\begin{array}{ll} \text{Minimize} & \alpha + \beta \\ \text{such that} & k\alpha + (1-k)(1-\beta) = \omega(1-k) \end{array}$$

It is easy to derive the false accept or false reject:

$$\begin{aligned} \alpha &= \sum_{s \in U} D(s)A(s) & (2) \\ \beta &= \sum_{s \in U} (1 - D(s))Q(s). & (3) \end{aligned}$$

Given Q, A, k and ω, we want to find the optimal policy, D^*, such that the effectiveness of the attack (1) is minimized. It turns out that the optimal filter policy has a nice and simple form. Each $D^*(s)$ is either 1 or 0, except for one particular $D^*(s_j)$. The optimal policy can be computed as the following. Firstly we sort the features in the descending order of the ratio $(A(s)/Q(s))$, and label them as $\langle s_1, \ldots, s_n \rangle$, where $n = |U|$ is the total number possible features. Next we find a j such that the constraint is met exactly when $D^*(s_i) = 0$ for $i < j$, $D^*(s_i) = 1$ for $i > j$, and $D^*(s_j) \in [0, 1]$. It is not difficult to see that such j can always be found. If we ignore $D^*(s_j)$, the optimal policy is deterministic.

It is interesting to note that the optimal policy derived is similar to the use of Conditional Legitimate Probability in [5].

Some applications may want to minimize the total number of wrongly accepted and rejected packets. In this case an alternative effectiveness function $e = k\alpha + (1-k)\beta$ can be used. Since the rate control mechanism already ensures that only limited number of packets are accepted, the main concern is to get the legitimate packets passing through. Hence another alternative is to choose $e = \beta$. Interestingly, all of the above 3 choices of effectiveness function yield the same optimal policy. In this paper, we use the attack effectiveness defined in Equation (1) as a measure of the filter performance.

B. Limitations of Static Policy

The optimal policy D^* is designed for a known and static attack distribution $A = \widetilde{A}$. Suppose that the actual attack distribution is not $\widetilde{A}$, or the attack is not static, what would be the performance of the D^*?

1. The performance of the static filter relies on the assumption that the attackers do not know the nominal traffic profile Q. This can be argued that, in practice, it is difficult for an outsider to perform traffic analysis. However, it is also not fair to assume the attack distribution A is uniform. There are some commonly perceived network statistics. For example, certain port numbers (such as 80, 25) are more likely to appear than other values. In Section VII-B, we conduct experiments using traces of 2 different networks to illustrate that, using partial knowledge of Q (which is obtained by analyzing a trace on another network), the attacks could be effective.

2. It is desirable to design a policy D that performs well if the attack distribution A is the same as the distribution $\widetilde{A}$ perceived by the filter, and performs no worse than a random policy if the distributions are sufficiently different. By random policy, we refer to a policy D_0 that drops each packet with a fixed probability, without considering the content in the packet. That is, $D_0(s) = c$ for all features in U, where c is a constant. Unfortunately, there is no such policy D unless the nominal traffic profile Q and the policy D has the following special form, which is unlikely in practice.

- For any s, if $Q(s) > 0$, then $D(s) = c$, where c is a constant, otherwise $D(s) = 0$.

Hence, in order not to perform worse than a random policy, the filter has to hide D from potential attackers.

3. It may be possible for the attackers to probe the filter in order to learn sufficient information about D. Such probing is possible if the attackers get a feedback on whether an attack packet reaches the destination or not. While such feedback may not be possible for all packets, it is possible for a large class of packets, for example TCP packets during connection setup phase (3-way handshake) and data transfer (data and acknowledgements).

V. Dynamic Attackers and Adaptive Filter with No Feedback

A natural remedy to overcome the limitations of a static filter is adaptation. The filter may try to learn the attack distribution through measurements and adapt its policy based on the revised $\widetilde{A}$ learnt from the measurements. By using a more accurate estimate of the attack distribution, the policy can perform closer to the optimal policy. Such adaptive filter is desirable, if the attacker's behaviour is static, which is the basis for most statistical-based filtering policies against DDoS attacks. Let e_0 be the attack effectiveness when the attack commences, and e_1 be the effectiveness if the perceived attack is correct, and the attackers remain static. The goal of an adaptive filter is to learn the attack distribution and reach e_1 as soon as possible.

A naive attack method is as follows. Firstly, the attackers determine a good attack distribution A. When the attack commences, all packets are generated according to A. In this case, as the distribution A is revealed, the filter can quickly learn A and adapt to the corresponding optimal policy. However, the attackers have no obligation to stay static. They can purposely behave erratically to trick the filter, and reveal A slowly. A good attack strategy not only extends the learning time of the filter, but also achieves average effectiveness higher than e_1 during the learning period. Furthermore, even if the filter chooses to remain static, the average effectiveness of such attacks is the same as e_0.

A. Adaptive Filter

In our model, we consider the ideal filter that can analyze the traffic within a slot and correctly learn the attack distribution A in that slot. After knowing A, the filter can incorporate such knowledge in the design of the policy for the next slot. Here are two strategies that the filter could employ.

S1. At the beginning of the i-th slot, the filter uses the traces in W previous slots to determine $\widetilde{A}_i$, where W is the size of learning window. Next, with respect to $\widetilde{A}_i$ and Q, the filter derives the "optimal" policy. Let this policy be D. In the special case of $W = 1$, the filter only uses the trace in one previous slot. The filter may also choose to use all the previous traces to derive $\widetilde{A}_i$. In this case, we write $W = \infty$. We will show later that this strategy can be disastrous in the short term.

S2. Similar to the previous strategy, at the beginning of the i-th slot, the filter use the traces in W previous slots to determine $\widetilde{A}_i$. However, the filter does not directly derive the optimal policy based on the attack distribution learnt from the traces. Instead, it takes the perceived attack distribution $\widetilde{A}_i$ as $\widetilde{A}_i = \gamma A' + (1-\gamma)\widetilde{A}_1$, where γ is a parameter and A' is the learnt attack distribution. Next, with respect to $\widetilde{A}_i$ and Q, the filter derives the optimal policy. In the case when $\gamma = 1$, this is simply strategy S1. When $\gamma = 0$, there is no adaptation. We will show that while this weighted strategy can avoid the disastrous behaviour of S1, it is still vulnerable.

B. Erratic Attackers

Erratic attackers switch their behaviours frequently so as to trick the filter into learning the wrong profile. The following is one such attacker.

Preparation. Let A be the attack distribution the attackers believe is effective against a static filter. The attackers also decide on the number of slots, say m, they intend to trick the filter. The attackers then find a sequence of distributions $C_0, C_1, \ldots, C_{m-1}$ such that

- A is a mixture of C_i. That is, $A = (1/m)(C_0 + C_1 + \ldots + C_{m-1})$. In this equation, the notations A and C_i each refers to the p.d.f of the corresponding distribution.
- There is no overlap in any two different C_i and C_j. That is, if $C_i(s) > 0$, then $C_j(s) = 0$.

Attack Phase. The attack consists of cycles of m slots each. During the i-th slot of each cycle, the attackers generate a set of feature S according to C_i. Next, the attackers generate attack packets with features in S, such that each feature in S has equal number of corresponding packets. For a DDoS attack, the attackers will need loose synchronization of the slots. This should not be a problem since the slots are much larger than the average network round trip times (RTT). In the analysis, we will assume that the attackers are synchronized.

C. Analysis of Erratic Attacks

This attack is very effective against strategy S1 in the first cycle.

Assume that the filter correctly learns each C_i by analyzing the trace in the i-th slot, and that the perceived attack distribution $\widetilde{A}_{i+1}$ in the $(i+1)$-th slot is based on $C_0, C_1, \ldots, C_i$.

Note that the actual attack distribution in the $(i+1)$-th slot is C_{i+1}, which has no overlap with C_j for all $j \le i$. Now, if the filter uses the wrongly perceived attack to derive the optimal policy, the consequences is damaging. Under this "optimal" policy, all packets generated by C_{i+1} is likely to be accepted. On the other hand, packets that are deemed from $C_0, C_1, \ldots, C_i$ become more likely to be rejected. In the worst case, all packet s, where $C_j(s) > 0$ for all $j \le i$ could be rejected, and the attack packets generated from C_{i+1} are used to fill up the rate. Hence, the filter will perform even worse than the static filter.

Instead of using the correctly determined attack distribution (which could be a deception employed by the attackers), strategy S2 slows down the filter's reaction through incorporating older knowledge and intentional error. As a result, S2 avoids some of the problems of S1 and is less susceptible to the simple attack described above. However, as we will see in Section VII-C, filter that uses strategy S2 still performs worse than static filter during the learning period.

D. Robustness of Erratic Attacks

Erratic attacks are quite robust. In case where the filter does not employ adaptation and yet the attackers use the above strategy, the effectiveness is still same as that of static attackers. This is due to the fact that A is a mixture of the C_i's.

The attackers that we have described knows exactly when the filter decides to adapt and switch the policy. It is unreasonable to assume that the attackers know this information perfectly. Nevertheless, these attackers are still effective even if the slots are not perfectly synchronized with the filter. In addition, if the attackers have probing abilities, they can detect when the filter switches its policy.

VI. Attack with Feedback

In the previous section, we present how an attack can be effectively launched without any feedback on whether the attack packet succeeded in reaching the victim or not. In this section, we explore how allowing feedback can aid the attackers. Feedback is possible for certain types of packets. For example, the attackers just need to listen for acknowledgement during TCP 3-way handshake.

The feedback mechanism can be deployed to probe the filter. With probing, the following information of the filter can be obtained.

I1. Whether the filter carries out adaptation. If so, the time when the filter switches its policy.

I2. A good choice of features that are accepted by the filter with high probability.

This information can be used, for example, by erratic attackers (Section V-B) to synchronize the attacks with the filter switching, so as to maximize the effectiveness of the attacks. With I2, the attackers could, at the beginning of each slot, probe for a set of good features and flood the filter with the corresponding packets.

I1 can be obtained by straight forward monitoring. In this section, we will focus on I2.

The search of such features is complicated by the rate control mechanism of the filter. Without rate control, each attacker just need to generate sufficient packets to find one good feature s and next repeatedly send many packets with s. With rate control, a sudden increase in packets of feature s would trigger the filter to readjust its policy, and hence there is a possibility that s is dropped under the adjusted policy.

In this section, we give a simple attack algorithm that targets probabilistic policy without rate control, and an approach to handle rate control. An evaluation of the attacks using network trace will be presented in Section VII-D.

A. General Probing Algorithm

This probing algorithm targets at probabilistic policy. It attempts to efficiently find a set of features S^* that is accepted by the filter with high probability. The probing algorithm is divided into two rounds as follows.

- *First Round.* The attackers determine a good attack distribution A, and generate as many packets according to A as possible. Next, the attackers listen for acknowledgements. Let S be the set of features accepted by the filter.
- *Second Round.* For each feature $s \in S$, the attackers generate and send 25 packets that has the feature s. Let $a(s)$ be the number of packets accepted with feature s. Then the attackers sort S in the decreasing order of $a(s)$. The output S^* is the first T_0 features in the sorted order, where T_0 is some parameter.

The goal of the first round is to find potential candidates. The second round further tests each of these candidates using 25 more probes. Essentially, the second round attempts to estimate the probability $D(s)$ for each $s \in S$. By using 25 probes, the estimation has high level of confidence.

The issue of efficient probing can be formulated into an interesting algorithmic problem. One possible formulation is: *Using minimum number of probes, determine a set S^* of T_0 features, such that with 99% confidence, the average* $(1/T_0)\sum_{s\in S^*} D(s) > 0.5$.

The proposed probing algorithm is certainly not optimal under this formulation, and probably not optimal in other reasonable formulations. For example, we have a more sophisticated probing that uses slightly less probes (details omitted). Nevertheless, the proposed simple algorithm is at most a constant factor away from the optimal, and is sufficient for a successful attack.

B. Handling the Effect of Rate Control

The problem with the previous probing algorithm is that, once the attackers flood the filter with packets carrying features from S obtained from the second round, the rate control mechanism would likely be triggered, causing adjustment in the filter policy. Some features in S that used to be accepted would be dropped by the adjusted policy. As a result, the probing algorithm needs to be modified slightly. Note that whenever the filter policy is adjusted, if a packet with features s is dropped, under the optimal policy, all packets with feature s_0 such that $A(s_0)/Q(s_0) \geq A(s)/Q(s)$ will be dropped too. This observation allows the attackers to continuously search for a "better" feature in multiple rounds using the following algorithm.

Given a good attack distribution A, the attackers do the following steps.

1) Generate and send large number of attack packets according to A.
2) Update A to be the distribution of accepted packets.
3) Update S to be the set of accepted features.
4) Repeat from the first step until there are less than T_0 features in S, where T_0 is some parameter.

The effectiveness of this attack algorithm is illustrated by the evaluation results in Section VII-D.

VII. Evaluation

In this section, we evaluate the effectiveness of attacks, given various attacker and filter strategies. All evaluations are done in a stand-alone setting via simulations, while important data (such as the nominal traffic profile) are obtained by analyzing real Internet traffic traces.

We will give the overall settings for our simulations in Section VII-A, and show our results when the attackers are static (Section VII-B). When the attackers are dynamic and the filter is adaptive, we show the results when there is no feedback (Section VII-C), and when there is feedback (Section VII-D).

A. General Settings

1) Network Traces: In our simulations, two traces from the real Internet are used. The filter has access to only the first trace, from which the nominal traffic profile Q is derived. The attackers have access to only the second trace, from which an approximation Q' of the nominal traffic profile is derived and is used to launch attacks.

- **Trace I**: A 15 minute trace starting from 1400 hours, April 25, 2004. This trace was collected on a trans-Pacific line (18Mbps CAR on 100Mbps link), and is maintained by the MAWI Working Group of the WIDE Project (http://tracer.csl.sony.co.jp/mawi/samplepoint-B/2004/200404251400.html). There are about 8.6 million packets in the trace, almost all of which contain IPv4 datagrams.
- **Trace II**: A 10 minute trace starting from 0900 hours, August 14, 2002. This trace was

from the Abilene-I data set maintained by the Passive Measurement and Analysis (PMA) project (http://pma.nlanr.net/Traces/long/ipls1.html). The entire data set consists of a pair of two hour contiguous bidirectional packet header traces collected at the Indianapolis router node (ILLS). There are about 2.5 million IPv4 datagrams in the trace.

2) Packet Attributes: The attributes used in our simulation consist of (1) the IP header length, (2) total IP datagram length, (3) fragmentation, (4) time-to-live (TTL), (5) transport layer protocol type, (6) source IP prefix[1]. For datagrams that carry TCP headers, we also include (7) TCP header length, (8) TCP flags, and (9) the smaller of the source and destination port number as an approximation of the server port number.

3) Nominal Profile: The filter treats Trace I as the nominal traffic and estimates the nominal traffic profile Q. The filter assumes that all attributes are independent, and stores the histogram obtained from Trace I in iceberg-style, where only the most frequently occurred values are kept in the histogram, such that at least 95% of the entries from the trace are covered. Other values that are not in the histogram are assumed to have a fixed small probability of occurring.

4) Traffic Rate Control: Throughout our simulations, we assume that the incoming attack traffic rate is 10 times the nominal traffic rate.

Recall from Section III-E that the desired traffic ratio ω is the ratio of accepted traffic rate over incoming legitimate traffic rate. For our simulations, this ratio is always 2. That is, we want to accept as many as twice the number of legitimate packets.

Under rate control, the smallest achievable effectiveness is 0.1, where the false accept is 0.1, and the false reject is 0. This is the "ideal" filter that accepts all legitimate packets and uses the remaining rate to accept 10% of the attack packets. On the other hand, a filter that rejects all legitimate packets will have false reject 1 and false accept 0.2. Hence, the effectiveness varies between 0.1 to 1.2 and the attack effectiveness on a random filter is always 1.

As mentioned in Section III-B, we assume that the policy can be adjusted in 0 time to achieve the desired rate.

5) Application-Level Semantics: In practice, packets may have different importance. For example, dropping the initial SYN packets for a TCP session is much more disruptive than dropping a data packet later on. In the simulation, we do not take into account the effect of any transport or application level semantics.

6) Centralized Attacker: For simplicity, a single attacker is used in the description of the attacks. All the attacks presented can be easily implemented in a distributed fashion while requiring only minimum coordination in some cases.

B. Static Attacker

1) Attack Distributions: In our simulations, we assume that the attacker uses one of the following attack distributions to generate the attack packets. For all these distributions, the IP prefixes are uniformly generated.

a) $\mathcal{A}_1$ (uniform): All attributes are uniformly distributed over all possible values.

b) $\mathcal{A}_2$ (refined uniform): All attributes are uniformly distributed over some values that are believed to be common in most networks. These values are obtained in two steps. First, we analyze Trace II, and take the most frequently occurred values. Next, we make some reasonable guesses, for example, we add some popular port numbers as server port numbers, and remove those port numbers above 1024.

In particular, in our simulation the IP header length is always 20 bytes, the transport layer protocol type is always TCP, the IP datagram is never fragmented, TTL is centered below 255, 128, and 64, the IP datagram size is one of the 10 most frequently occurred values in Trace II, the TCP header length is always 5, the TCP flags are one of 2 (SYN), 16 (ACK) and 24 (ACK and PSH), and server port number is chosen from 10 values between 0 and 1023 which either occurs frequently in Trace II, or is used by a popular protocol (such as SSH, SMTP, FTP, HTTP, etc).

c) $\mathcal{A}_3$ (guessed nominal): All attributes follow the distribution Q', except for source IP prefixes, which are uniformly chosen. That is, ignoring source IP addresses, the attack packets statistically reassemble the packets in Trace II. Note that the attacker does not need to know $\mathcal{A}_3$ explicitly. To generate attack packets, the attacker can just use samples from Trace II. This gives the attacker two advantages. Firstly, traffic analysis is not necessary. Secondly, some hidden statistical properties will still be preserved in the attack packets.

2) Static Filter: Let D_1,D_2 and D_3 be the optimal policies when the perceived attack distribution is $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}_3$ respectively.

In Table I we illustrate the effectiveness of attacks for various attack distributions and filter policies, where both sides are static.

We can see that if the perceived attack distribution happens to be the same as the actual attack distribution ($\mathcal{A}_1$ vs D_1, etc), the filter can be near optimal, i.e., $\alpha \approx 0.1$ and $\beta \approx 0$.

However, if the perceived attack distribution is not the same as the actual attack distribution, then the performance of the filter can be quite bad. For example, when the filter assumes that attacks are uniformly distributed and employ policy D_1, and the attacker happens to use attack distribution $\mathcal{A}_2$ or $\mathcal{A}_3$, then the attacker achieves effectiveness above 0.17, or false reject more than 5%.

Interestingly, if the filter thinks that the attack packets are from $\mathcal{A}_2$ and employs policy D_2, and it happens that the attacker actually uses a simple uniform distribution $\mathcal{A}_1$, then the attacker can achieve a false reject of almost 50%. Similar situation happens when attacker uses $\mathcal{A}_3$. This seems to indicate that if the filter tries to make a smart guess and is unfortunately wrong, then its performance can be very bad, no matter the attacks are of a simpler form, or a more complicated form.

[1]Similar to [5], only the first 16-bit of the IP addresses are kept in the histogram, as an approximation of the network of the IP address.

TABLE I

STATIC ATTACKER AND FILTER

	D_1	D_2	D_3
$\mathcal{A}_1$	$\alpha = 0.1109$ $\beta = 0.0000$ $e = 0.1109$ $\omega = 2.1089$	$\alpha = 0.1453$ $\beta = 0.4964$ $e = 0.6416$ $\omega = 1.9562$	$\alpha = 0.0874$ $\beta = 0.0000$ $e = 0.0875$ $\omega = 1.8744$
$\mathcal{A}_2$	$\alpha = 0.1273$ $\beta = 0.0574$ $e = 0.1848$ $\omega = 2.2161$	$\alpha = 0.0904$ $\beta = 0.0000$ $e = 0.0905$ $\omega = 1.9044$	$\alpha = 0.1110$ $\beta = 0.0122$ $e = 0.1232$ $\omega = 2.0978$
$\mathcal{A}_3$	$\alpha = 0.1034$ $\beta = 0.0709$ $e = 0.1743$ $\omega = 1.9635$	$\alpha = 0.1527$ $\beta = 0.4980$ $e = 0.6507$ $\omega = 2.0293$	$\alpha = 0.1039$ $\beta = 0.0000$ $e = 0.1039$ $\omega = 2.0387$

3) Adaptive Filter: If the filter is adaptive and learns the attack distribution, it can perform optimally, provided that the attacker remains static. The diagonal entries in Table I show the optimal performance.

C. Dynamic Attacker and Adaptive Filter with No Feedback

1) Simulation Slots: The simulations are divided into *slots* as mentioned in Section III-B. In each slot, the attacker chooses a distribution from which the attack packets are generated, and the filter chooses a perceived distribution, which may be based on the observed attack distributions in the previous slots, and derives the optimal dropping policy. In our simulations, 40 slots are simulated, in each of them 10^5 attack and legitimate packets are generated.

We assume that the filter has the ability to know the exact distribution used by the attacker in every previous slot. As mentioned in Section III-E, the filter may choose to keep the history of attack distributions for W slots. In the simulations, W ranges from 0 to 40.

2) Attack Distribution: Based on an initial guessed nominal traffic profile Q', an erratic attacker derives the following distributions.

$\mathcal{A}_4$ *(erratic guessed nominal):* Similar to $\mathcal{A}_3$, but for each attribute, the distribution is equally divided into M pieces, where M is a parameter chosen by the attacker. For illustration, we will use $M = 10$ in our simulations unless otherwise stated. For example, suppose the distribution for the transport layer protocol type is B, which is formed by 95% TCP and 5% UDP, then after division we would have 10 distributions $B_1, \ldots, B_{10}$, where $B_1, \ldots, B_9$ are the same, which are 100% TCP, and B_{10} consists of 50% TCP and 50% UDP.

When a new attack distribution is to be determined, one piece of distribution for each attribute is randomly chosen from the M pieces, and the attack packets are then generated according to the chosen pieces.

Since the filter can accurately learn each B_i, when the learning window size W is larger than M, the filter can learn Q'. This is verified in Fig. 2, which shows a steady decrease in e when $W > 10$.

3) Filter with Strategy S1: In Fig. 2, we illustrate how the effectiveness changes with the filter's learning window size. The solid lines show the average false accept, false reject, and effectiveness over the 40 slots. For example, when $W = 10$, the average effectiveness over 40 slots is about 0.3. The dashed lines show e_1, the optimal effectiveness when the perceived attack distribution is actually Q', and e_0, the effectiveness when no adaptation is carried out. We assume that the filter uses strategy S1 (Section V), that is, the learnt attack distribution in the previous W slots is used as the perceived attack distribution in the current slot to derive the optimal policy. The standard deviations of the false accept and false reject are shown in Fig. 3.

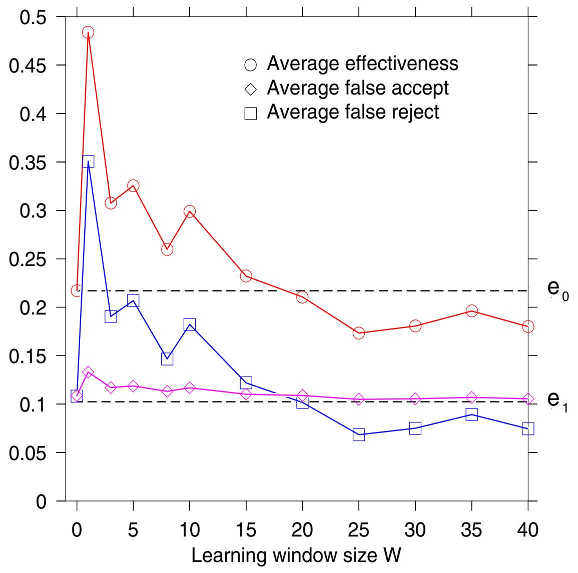

Fig. 2. Erratic attacker and adaptive filter with strategy S1. The average is taken over the 40 simulation slots. For example, when $W = 5$, the effectiveness 0.3255 is the average of the corresponding values as in Fig. 4.

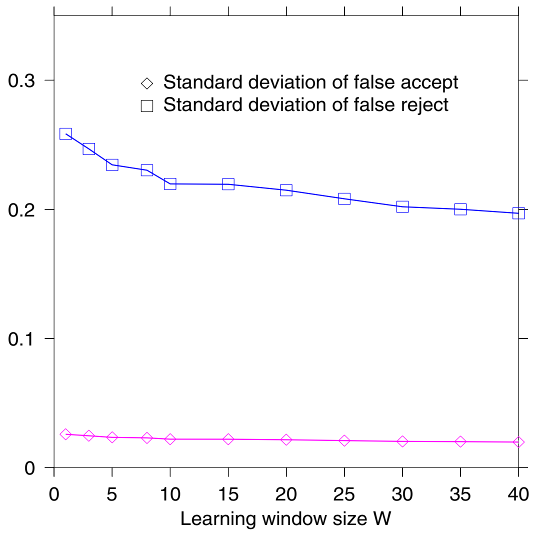

Fig. 3. Standard deviations of false accept and false reject with strategy S1.

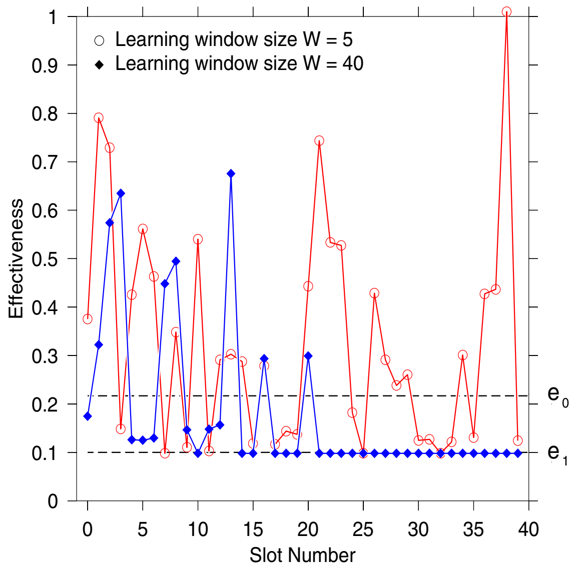

Fig. 4. Effectiveness when $W = 5$ and $W = 40$ ($\gamma = 0$)

We can see that if the size of the learning window is small (say, less than 10), then the attacks are quite effective, in the sense that the average effectiveness would be quite far from that in the perfect (static) case. In particular, when learning window is 1, the average false reject can go as high as 35%, while the false accept is about 14%. Only when the learning window size is above 20 before the average effectiveness becomes smaller than the case where $W = 0$, which means adaptation with a learning window size smaller than 20 would perform worse than static filtering. Notice that the false reject β never falls below 5%.

This is again verified in Fig. 4. In this figure we further illustrate the effectiveness over the 40 slots, for the cases when the learning window size is 5 and 40.

When the filter's learning window is 5, the changes in effectiveness in different slots is very drastic. We can see that in some slots, the effectiveness goes even beyond 1, which means that the filter is doing worse than random dropping in those slots.

When the filter has a large learning window ($W = 40$), the effectiveness still fluctuates in the first few slots because the filter has not learnt all the pieces of Q' yet. After 10 slots the variance in the effectiveness becomes smaller, but it is only after 20 slots that the filter learns all of Q' and achieves the optimal value of e_1.

Therefore, we can see that when the learning window is small, the filter perform very badly even on the average. When learning window is large, the filter can eventually learn the attack distribution and push the effectiveness close to the optimal e_1. However, during the learning process, especially the first few slots, the attack is still effective.

4) Filter with Strategy S2: To overcome these disadvantages when using strategy S1, the filter could employ the second strategy S2 (Section V), where the perceived attack distribution is not what is learnt in the previous slots, but a weighted composition of what is learnt and the initial guess (which assumes a uniform distribution).

Fig. 5 illustrates the improvement of S2 over S1. Fig. 6 shows the corresponding standard deviations. In Fig. 5, the filter uses $\gamma = 0.5$. That is, the perceived attack distribution $\widetilde{A}$ is the weighted composition of the learnt attack distribution in the previous W slots, and the uniform distribution, where both distribution have equal weight. Similar to Fig. 2, the dashed line represents the perfect (static) case where the filter, using strategy S2, knows the distribution Q' and achieves the effectiveness of e_1. When we compare Fig. 5 with Fig. 2, it is clear that by using strategy S2, the average effectiveness of the attacks is reduced drastically when the learning window W is small, compared to the case where strategy S1 is used.

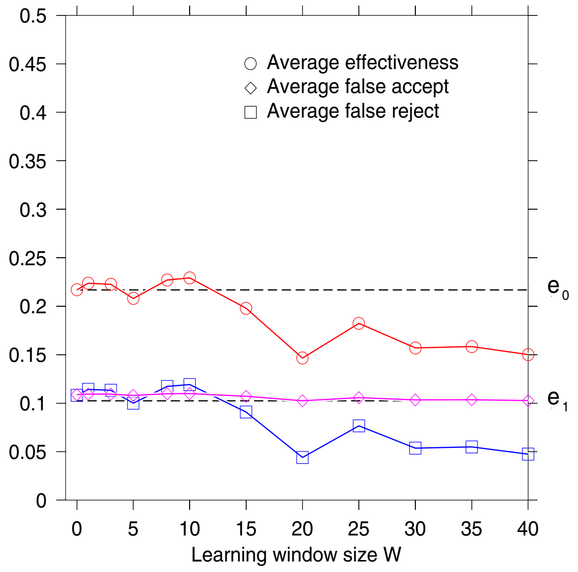

Fig. 5. Erratic attacker and adaptive filter with strategy S2 ($\gamma = 0.5$)

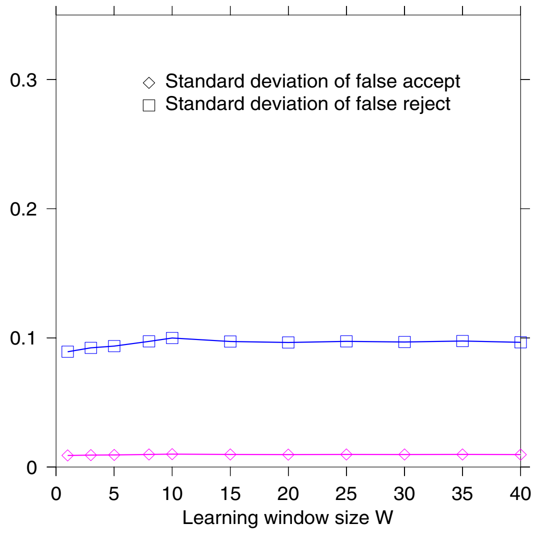

Fig. 6. Standard deviations of false accept and false reject with strategy S2.

Note that the optimal effectiveness here is almost the same as that in Fig. 2. This shows that mixing the learnt attack distribution with a uniform distribution would, asymptotically, not make much lost in performance, but significantly reduce the average attack effectiveness when the learning window is small.

On the safe side, the filter should learn slowly and one way to achieve that is to purposely introduce noise in the learnt distribution. Nevertheless, this strategy still performs worse than static filtering during at least the first 10 slots.

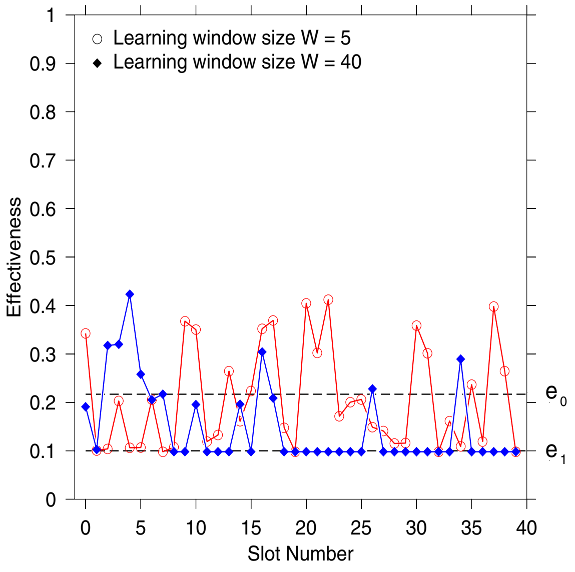

Fig. 7. Effectiveness when $W = 5$ and $W = 40$ ($\gamma = 0.5$)

Fig. 7 shows the attack effectiveness for 40 slots, where $\gamma = 0.5$ and the learning window size is 5 and 40. As we can see, the variance of the effectiveness in both cases is also reduced compare to S1. However, there is a price to be paid for using S2. When S2 is used, even after a long learning period, the attack effectiveness can still spiked up to a much larger value.

We have also done additional simulations with various values of M. The results confirm that as M becomes larger, it takes longer for the filter to learn the attack distribution, and its performance is worse than static filtering during the learning process. We compare the effectiveness of the attacks for $M = 5$, $M = 10$ and $M = 20$ in Fig. 8 and 9, where $\gamma = 0$ and $\gamma = 0.5$ respectively.

D. Dynamic Attacker with Feedback

In this section, we illustrate the effectiveness of the attacks with feedback by simulating the probing algorithm presented in Section VI. The result is shown in Fig. 10 .

Recall from Section VI that the attacker can make use of feedback to make good choices of features that will be accepted by the filter with high probability. With the rate control mechanism, it is not possible to make the false accept to go higher than 0.2, but as we can see in Fig. 10, the attacker can force the false reject to go unacceptably high.

In our experiment, we assume that the attacker initially generates 10^5 packets according to Q' with uniform source

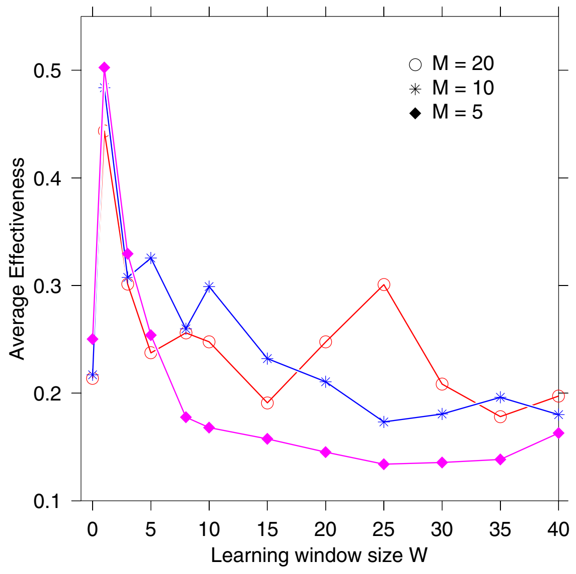

Fig. 8. Average effectiveness for different M ($\gamma = 0$)

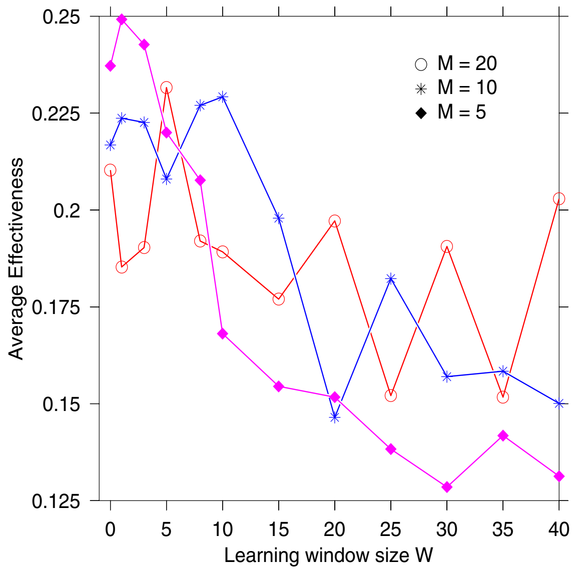

Fig. 9. Average effectiveness for different M ($\gamma = 0.5$)

IP prefixes. We assume that the attack distribution perceived by the filter is uniform. In the first round, the attacker sends all the packets and see which packets are accepted. In the subsequent rounds, the attacker learns the distribution of the features of those accepted packets and send according to the learnt distribution. The total number of attack packets in each round is always maintained at a rate that is 10 times the nominal traffic rate. We can see that after the 5-th round, the false reject β gets very close to 1 and the false accept α gets close to 0.2. Hence, attackers with feedback can force the performance to be worse than random filtering.

In this case, one possible way to prevent the attacker from learning from the feedback is to keep the initial false accept α as close to 0.1 as possible. However, this is very difficult

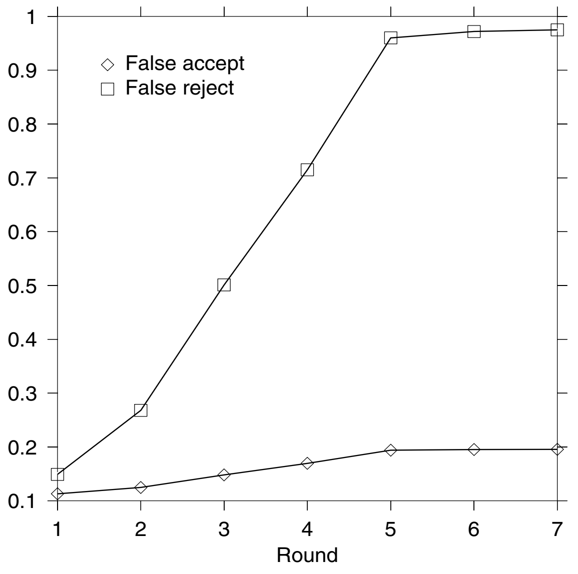

Fig. 10. Dynamic attack with feedback

to achieve in practice.

VIII. Conclusion

There is a growing interest in the use of statistical-based filtering to identify "abnormal" packets during DDoS attacks. As such statistical techniques gain popularity, DDoS attacks with the corresponding counter-measures should be expected to appear in the future. In this paper, we study the effectiveness of statistical-based filtering against possible counter-measures. We show that optimal static filters are not robust in the sense that, if a filter wrongly estimates the attack distribution, its performance would be far from optimal. Although adaptive filters seem to be a potential remedy, we also show that they can perform much worse than a static filter, when the attackers behave in an erratic manner such that it is difficult for the filter to adapt its policy effectively. Adaptive filters are also vulnerable to attackers with probing ability, or have partial knowledge of the network statistics, if they are unable to adapt much faster than the attackers. Although our results are pessimistic, there are ways to enhance statistical-based defence. For example, effective attacks often require that initial effectiveness cannot be too low. It may be possible to use application-layer knowledge, in addition to network and transport layer knowledge, to significantly reduce the initial attack effectiveness.

References

[1] Jelena Mirkovic, Gregory Prier, and Peter L. Reiher, "Attacking DDoS at the source," in *Proceedings of the 10th IEEE International Conference on Network Protocols*, September 2002, pp. 312–321.

[2] L. Feinstein, D. Schnackenberg, R. Balupari, and D. Kindred, "Statistical approaches to DDoS attack detection and response," in *Proceedings of the DARPA Information Survivability Conference and Exposition*, April 2003, pp. 303–314.

[3] Cheng Jin, Haining Wang, and Kang G. Shin, "Hop-count filtering: An effective defense against spoofed traffic," in *Proceedings of the ACM Conference on Computer and Communications Security*, 2003, pp. 30–41.

[4] Tao Peng and K. Ramamohanarao C. Leckie, "Protection from distributed denial of service attacks using history-based IP filtering," in *Proceedings of the IEEE International Conference on Communications*, May 2003, vol. 1, pp. 482–486.

[5] Yoohwan Kim, Wing Cheong Lau, Mooi Choo Chuah, and Jonathan H. Chao, "PacketScore: Statistics-based overload control against distributed denial-of-service attacks," in *Proceedings of IEEE INFOCOM*, 2004, pp. 2594–2604.

[6] Jelena Mirkovic, *D-WARD: Source-End Defense Against Distributed Denial-of-Service Attacks*, Ph.D. thesis, UCLA, 2003.

[7] J. B. D. Cabrera, L. Lewis, X. Qin, W. Lee, R. K. Prasanth, B. Ravichandran, and R. K. Mehra, "Proactive detection of distributed denial of service attacks using mib traffic variables-a feasibility study," in *Proceedings of the IEEE/IFIP International Symposium on Integrated Network Management*, May 2001, pp. 609–622.

[8] Haining Wang, Danlu Zhang, and Kang G. Shin, "Detecting SYN flooding attacks," in *Proceedings of IEEE INFOCOM*, 2002, vol. 3, pp. 1530–1539.

[9] Ratul Mahajan, Steven M. Bellovin, Sally Floyd, John Ioannidis, Vern Paxson, and Scott Shenker, "Aggregate-based congestion control," *Computer Communication Review*, vol. 32, no. 3, July 2002.

[10] Kihong Park and Heejo Lee, "On the effectiveness of route-based packet filtering for distributed DoS attack prevention in power-law internets," in *Proceedings of ACM SIGCOMM*, 2001, pp. 295–306.

[11] Stefan Savage, David Wetherall, Anna Karlin, and Tom Anderson, "Practical network support for IP traceback," in *Proceedings of ACM SIGCOMM*, 2000, pp. 295–306.