Detecting Service Violations and DoS Attacks

Ahsan Habib, Mohamed M. Hefeeda, Bharat Bhargava
CERIAS and Department of Computer Sciences
Purdue University, West Lafayette, IN 47907-1398
{habib, mhefeeda, bb}@cs.purdue.edu

Abstract

Denial of Service (DoS) attack is a serious threat for the Internet. DoS attack can consume memory, CPU, and network resources and damage or shutdown the operation of the resource under attack (victim). A common DoS attack floods a network with bogus traffic so that legitimate users may not be able to communicate. There are several proposals to *traceback* the network attack path to identify the source that causes the DoS attack. This is an effective solution to trace the attacker but it is not preventive in nature. *Ingress filtering* and *Route-based filtering* are two proactive approaches to stop DoS attacks. These solutions check source addresses of incoming packets to ensure they are coming from legitimate sources or traversing through proper routes. We study several existing schemes that deal with DoS attacks. We describe several network monitoring approaches to detect service violations and DoS attacks. In addition, we propose a new distributed scheme to reduce monitoring overhead. Finally, a quantitative comparison among all schemes is conducted, in which, we highlight the merits of each scheme and estimate the overhead (both computation and communication) introduced by it. The comparison provides guidelines for selecting the appropriate scheme, or a combination of schemes, based on the requirements and how much overhead can be tolerated.

I. Introduction

INTERNET security lapses have cost the U.S. corporations about 5.7 percent of their annual revenue as reported by economist Frank Bernhard from University of California at Davis [1]. San Diego Supercomputer Center reported 12,805 denial of service (DoS) attacks over a three-week period in February 2001 [2]. These attacks can be severe if they last for a prolonged period of time preventing legitimate users from accessing some or all of computing resources. Imagine an executive of a financial institution deprived of access to stock market updates for several hours or even several minutes. In [2], the authors showed that whereas 50% of the attacks lasted less than ten minutes, unfortunately, 2% of them lasted greater than five hours and 1% lasted more than ten hours. There were dozens of attacks that spanned multiple days. Wide spectrum of motivation behind these DoS attacks exists. They range from political conflicts and economical benefits for competitors to just curiosity of some computer geeks. Furthermore, cyber terrorism may not be excluded in the future.

The aim of a DoS attack is to consume the resources of a victim or the resources on the way to communicate with a victim. A victim can be a host, server, router, or any other kind of entity connected to a network. The attack hinders many users/clients to contact the victim. Numerous security flaws in the existing systems make them vulnerable to DoS attacks. Even though many available cryptographic protocols can *theoretically* provide the desired level of security, they impose excessive overhead that might degrade the performance of the system. In addition, the software implementation of these protocols can never be guaranteed to be free of bugs. Inevitable human errors during software development, configuration, and installation open several unseen doors for attacks. Several DoS attacks are known and documented in the literature [2], [3], [4], [5]. Flooding the victim's network with overwhelming amount of traffic is the most common. This unusual traffic clogs the communication links and thwarts all communications among legitimate users. This kind of attacks may

CERIAS Tech Report TR 2002-15. Submitted to IEEE Network for review

result in shutting down an entire site or a branch of the network. This happened in February of 2000 for popular web sites such as Yahoo, E*trade, Ebay, and CNN for several hours [3]. TCP SYN flooding is an instance of the flooding attacks. Under this attack, the victim is a host and typically runs a Web server. A Web client usually sends a request for a file to the Web server using TCP SYN packet. The attacker sends a TCP SYN pretending a desire to establish a connection making the server reserve buffer for it. The attacker does not complete the connection. Instead, it issues more TCP SYNs, which lead the server to naively wasting its memory for *never-completed* connections. Sending such SYN requests with a high rate consumes the server's memory and makes it unable to satisfy connection requests from legitimate users. This is an example of denying services from the users. Other types of flooding includes TCP ACK or RST flooding, ICMP and UDP echo request flooding, and DNS request flooding [2], [4]. This list is by no means exhaustive. Usually, the attacker does not use the real IP address of his/her own machine, rather, s/he spoofs the source address of the attacking packets. This makes it harder to trace down the attacker.

DoS attack can be more severe when an attacker uses multiple hosts over the Internet to attack a victim. To achieve this, the attacker usually compromises many hosts and deploys attacking agents on them. The attacker signals all agents to launch an attack simultaneously on a victim. This attack is known as Distributed DoS (DDoS) attack. In this case, even if it is possible to trace the source(s), it is more difficult to trace the real attacker. Barros [6] shows that DDoS attack can reach a high level of sophistication by using *reflectors*. A reflector is like a mirror that reflects light. In the Internet, many hosts such as Web servers, DNS servers, and routers can be used as reflectors. The servers always reply to a SYN request in response to a query. The routers send ICMP packets (time exceeded or host unreachable) in response to particular IP packets. The attackers can abuse these reflectors to launch DDoS attacks. For example, an attacking agent sends a SYN request to a reflector specifying the victim's IP address as the source address of the agent. The reflector, without knowing this, will send a SYN ACK to the victim. There are millions of reflectors in the Internet and any attacker can use these reflectors easily to flood the victim's network by sending large amount of packets. Paxson [7] analyzes several protocols and applications and concludes that DNS servers, Gnutella servers, and TCP-based servers such as Web servers are potential reflectors. We discuss more on DDoS attacks and the mechanisms to detect and prevent them later in this paper.

Another threat facing computer networks, especially quality of service (QoS) enabled networks such as Differentiated Services (DS) networks, is the QoS attacks. In this setting, the attacker is a regular user of the network trying to get more resources (better service class) than what s/he has signed (paid) for. A QoS network provides different classes of service for different cost. Differences in charging models of the service classes can entice attackers to steal bandwidth and other network resources. Such attacks make use of known vulnerabilities in firewall filter rules to inject traffic or spoof the identity of valid customers with high QoS. Since, the DS framework is based on aggregation of flows into service classes, valid customer traffic may experience degraded QoS as a result of the injected traffic. Taken to an extreme, the attacks may result in a denial of service. This creates a need for developing an effective defense mechanism that can automate the detection and reaction to attacks on the QoS-provisioned network domain. QoS attacks are classified into two kinds: attacking the *network provisioning process* and attacking the *data forwarding process*. Network provisioning involves configuration of routers in a QoS network. This process can be attacked by injecting bogus configuration messages, modifying the content of real configuration messages, and delaying or dropping such messages. Networks can be secured against such attacks by employing encryption of the configuration messages of the signaling protocols. Attacks on the data forwarding process are of a more serious in nature. This attack involves injecting traffic into the network with the intent to steal bandwidth or to cause QoS degradation by causing other customer flows to experience longer delays, higher loss rates and lower throughput. Habib, et al [8] devise a network monitoring mechanism to detect attacks in QoS domains. This mechanism measures the Service Level Agreement (SLA) parameters such as delay, loss, and throughput and compares these measurements with the negotiated values between service provider and user. Employing this monitoring technique, a service provider can detect any service violation within its network domain. Additionally, the monitor can check whether excessive flows passing through its domain are destined to a particular network domain or not. This aggregation may cause DoS attacks targeted to its domain or to any other downstream domains.

In this paper, we discuss several techniques to traceback the attacker who causes DoS attacks. For each technique, we describe how an attack happens and the procedure to detect it. The limitations of each traceback procedure are analyzed. The traceback is an aftermath solution. Filtering mechanism can be used to protect networks against DoS attacks. Ingress, egress, and route-based filtering are discussed in this paper. For QoS-enabled network domains, continuous monitoring is necessary to detect service violations and bandwidth theft attacks. This violation detection enables the provider to detect DoS attacks. We discuss stripe-based and distributed monitoring schemes to defeat these attacks. We provide a quantitative comparison among several schemes used to deal (detect/prevent) with DoS attacks. We highlight the merits of each scheme and estimate the overhead (both computation and communication) incurred by it. This comparison helps to decide on selecting the appropriate scheme based on requirements.

II. DoS Attacks: Detection and Prevention

In this section, we discuss the approaches to detect and prevent DoS attacks using traceback and filtering. The ways to detect and prevent DoS attacks are shown in Figure 1. There are several ways to detect the source that causes a denial of service (DoS) attack. IP traceback is one of them. IP traceback can be done using either ICMP traceback messages or marking packets at the routers. The marking strategies at the routers can be of a deterministic and probabilistic type. Hash-based IP traceback provides source path isolation engine (SPIE) to track attackers even for low volume of packets. Monitoring a network can help to detect DoS attacks. One obvious way of monitoring is to log packets at various points of a network domain. For a QoS network, service level agreement (SLA) violation detection, discussed in Section III, can help to detect bandwidth theft and DoS attacks. It is to note that traceback is a detection approach rather than to prevent the attack. Filtering spoofed packets prevents networks from DoS attacks. Ingress/Egress filtering and route-based filtering are two approaches to prevent DoS attack in the Internet.

We use Figure 2 to demonstrate different ways to launch attacks and take actions against them. In Figure 2, hosts Hs are connected to domains Ds, which are connected to the Internet cloud. When a host sends packets to other hosts in the Internet, it travels through several routers Rs on its way. These routers are potential candidate to help in detecting and preventing DoS attacks. In Figure 2, A represents an attacker and V is the victim.

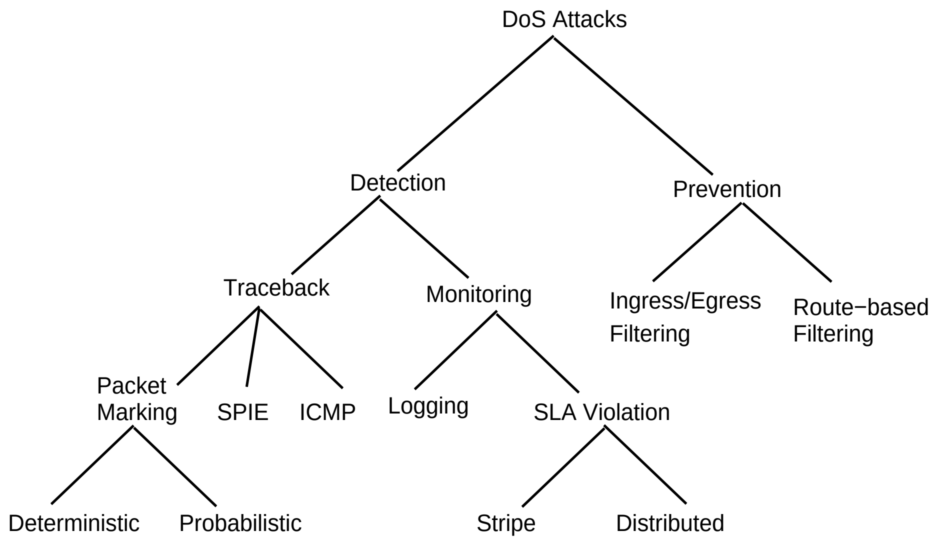


Fig. 1. Classification of approaches to detect DoS attacks and service violation.

A. Traceback of DoS Attack

Traceback is an effective scheme to determine the attacking source. It is difficult to trace the attack source because the attacker often spoofs the source IP address. In addition, the Internet is stateless, which means whenever a packet passes through a router, the router does not store any information (traces) about that packet. When a host sends a packet to another host over the Internet, it travels through several routers on its way and we can trace the network path that the attack traffic follows. We discuss several existing studies on how to traceback the attack source when any attack happens.

ICMP Traceback. Bellovin proposed ICMP traceback messages [6], where every router samples the forwarding packets with a very low probability (1 out of 20000) and sends an ICMP Traceback message to the destination. This message contains the previous and next hop addresses of the router, timestamp, part of the traced packet, and authentication information. In Figure 2, while packets are traversing network paths from attacker $A1$ to the victim V, the intermediate routers, Ri, sample some of these attack packets and send ICMP traceback messages to the destination, V. With enough ICMP traceback messages, the victim later can trace the network path, $V - A1$. This work shows a promising solutions for constructing path from victim to the source involved in attacking. The disadvantage of this approach is that sometimes ICMP packets can be ignored at routers and these traceback packets can be dropped. The attacker/source can defeat the authentication mechanism by sending many false ICMP traceback messages to confuse the victim, since the routers send only few messages.

To address Distributed DoS (DDoS) attack by reflectors, Barros [6] proposes a modification of ICMP traceback messages. In his refinement, routers sometimes send ICMP messages to the *source* of the currently being processed packet rather than its destination. In Figure 2, $A3$ sends a SYN request to $H3$ specifying V as the source address of this packet. $H3$ sends a SYN ACK to the victim V. According to the modification, routers on the path $A3 - H3$ will send ICMP messages to the victim. This *reverse trace* enables the victim to identify the attacking agent(s) from these trace packets. The *reverse trace* mechanism is helpful for defending against DDoS attacks by reflectors and depends only on the number of attacking agents rather than the number of reflectors [7]. This achieves scalability because number of available reflectors is much higher than number of attacking agents on the Internet.

Packet Marking at Routers. Burch and Cheswick [9] proposed to mark packets by inscribing the IP address of the routers in the header of the data packet themselves, that is no separate message is issued from the routers. The goal of this marking is that, after an attack, the victim can reconstruct the network path of an attack using information in the marked packets with high probability. This marking can be deterministic or probabilistic. In the deterministic marking, a router marks all packets and the packets are marked at all routers. The obvious drawback of deterministic packet marking is that it may require large packet header that grows with the increase of number of hops along the path. The overhead on routers will increase to mark every packet. The probabilistic packet marking encodes the local path information with a probability $p \ll 1$ in the packet header. During flooding attack, huge amount of traffic travels towards the victim. Therefore, there is a great chance that many of these packets will be marked at routers throughout their journey from the source to the victim. It is likely that the marked packets will give enough information to trace the network path from the victim to the source of the attack. In Figure 2, the attack traffic travels through different routers from $A1$ to V and the routers, Ri, inscribe local path information on the attack packet header. The victim can reconstruct the path from $V - A1$ provided that sufficient packets are sent from the attackers.

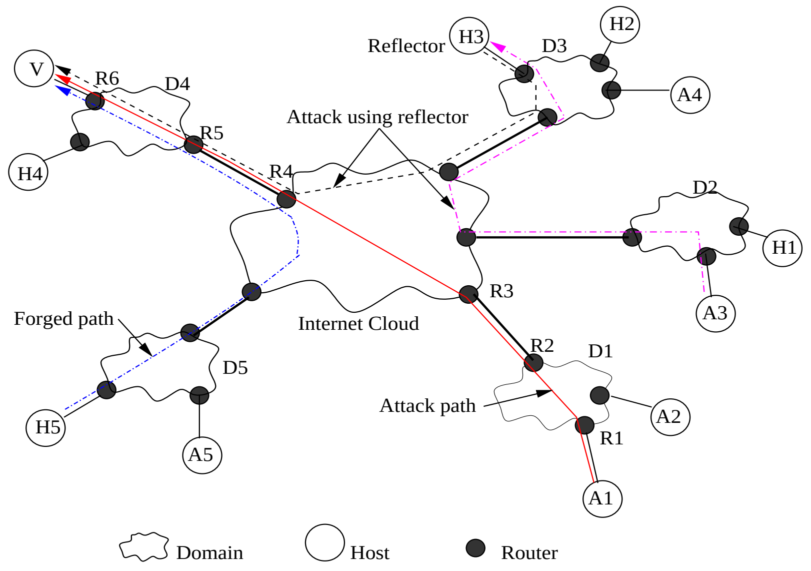


Fig. 2. Different scenarios for DoS attacks. Attacker, $A1$, launches attack to the victim V. $A1$ spoofs IP address of host $H5$ from domain, $D5$. $A3$ uses hosts $H3$ as a reflector to attack the victim. Packets travel through different routers, Ri, which are potential candidates to detect and prevent DoS attacks.

The probabilistic packet marking (PPM) and traceback are studied in details by Savage, et al [5]. The authors [5] describe efficient ways to encode addresses and distance metric on the packet to be marked. The distance metric represents how many hops the attacker is away from the victim. In Figure 2, if there is only one hop between routers $R3$ and $R4$, the attacker, $A1$, is 6 hops away from the victim, V. Suppose, Router $R1$ marks a packet and sets the counter to zero. On the way of the packet's trip, every router increments the distance metric. Thus, the victim knows the possible minimum distance of the attacker in terms of hops. Even though it is not impossible to reconstruct an ordered network path using an unordered collections of router samples, it requires the victim to receive a large amount of packets. To solve this problem, the authors [5] suggest to encode *edges*, start and end routers of a link, in the attack path rather than using individual routers. Using distance metric and edges together, it is easy to reconstruct the attack path with reasonable amount of packets received by the victim. To encode edges, 72 bits (two 32 bits IP addresses and one 8 bits distance metric) are required on the IP header. Instead of using two spaces for IP addresses, they use XOR (exclusive or) of two addresses to save space. For example, in Figure 2, router $R1$ encodes the XOR of $R1$ and $R2$ on the packet and sets the distance metric to zero. Other routers on the path just increase the distance metric of this packet, if they don't decide to mark it again. When this packet reach the victim, it provides a tuple of address and distance $< R1 \oplus R2, 5 >$. Some packets will be marked at router $R5$ and will provide a tuple $< R5 \oplus R6, 1 >$ to the victim. Some packets will be marked by the last router to provide $< R6, 0 >$ because there is no more router after that. As $R6 \oplus R5 \oplus R6 = R5$, it is possible to retrieve all routers on the path by XORing collected messages at the victim sorted by distances. The required space to mark a packet can be reduced by fragmenting the address information (XORed value) into some number k of smaller non-overlapping fragments. When a router decides to mark a packet, it uses one of the fragments and inscribe it into the packet. If enough packets are sent by the attacker, the victim will receive all fragments to extract all routers along the path. The authors [5] propose to use identification field (currently used for IP fragmentation) of IP header to encode the distance metric, fragment of edge, and offset of the fragment. Some solutions are described in [5] to cope with co-existence of marking and fragmentation of IP packets. This approach can reconstruct most network paths with 95% certainty if there are about 2000 packets available to traceback and even the longest path can be resolved with 4000 packets. For DoS attacks, this amount of packets is very much obtainable because the attacker needs to flood the network to cause DoS attacks. Moore, et al report that some severe DoS attacks had a rate of thousands of packets per second [2].

To devise a better packet marking algorithm, Song and Perrig [10] use network topology information to compress the representation of the edge state. For example, if the victim V knows the Internet topology graph connected to domain $D4$, all routers along the path to the victim do not need to encode the actual IP address to mark a packet. Instead, the routers can mark packets with hashed values of the IP addresses. The victim has the topology tree and can traverse the tree to extract IP address from the hashed values. This tree traversal enhances the path recovery scheme to detect distributed DoS attacks and improves the robustness of the whole marking scheme for trace-backing. Dean, et al [11] model the traceback problem as a polynomial reconstruction problem and use techniques from algebraic coding theory to provide robust methods of reconstructing the network path.

Snoeren, et al [12] propose a hashed based IP traceback technique that uses source path isolation engine (SPIE). The SPIE generates audit trails of traffic and can trace origin of single IP packet delivered by a network in recent past. The SPIE uses a very efficient method to store the information that a packet traversed through a particular router. To achieve this, n bits of the hashed values of the packet is used to set an index of a 2^n-bit *digest table*. When a victim detects an attack, SPIE samples all digest tables and stores it to query about the path of attack packets. Topology information is used to construct a graph from the victim to the neighbor networks. The SPIE simulates a reverse-path flooding from the victim by examining the digest table to check the path existed at the time the packet was forwarded. Querying the routers along the paths from the victim eventually reconstructs the attack path. The SPIE needs to enquire the routers to sample digest tables soon enough after the packet was forwarded by a router. Otherwise the record of the packet at the router will be replaced by record of new packets. The main advantage of SPIE over ICMP traceback messages and PPM is that SPIE can traceback the attack path even for low volume packets received at the victim.

Limitations of Probabilistic Packet Marking. The Probabilistic Packet Marking (PPM) has potential weaknesses because PPM is vulnerable to spoofing of marking field in the IP header by the attackers [13]. It is not certain that all attack packets will be marked by any of the intermediate routers. If each router has a marking probability p and there are h hops from the attacker to the victim, then the probability that a packet reaches the victim without marking by any intermediate router is $(1-p)^h$. Many of the packets can reach the victim without being marked at all and these packets will provide confusing information to the victim about the original path of the attack. In Figure 2, $H5-V$ is a forged path because the attacker $A1$ uses $H5$'s IP address as its source address and this packet was not marked by any of the intermediate routers. This forged path might be confused as an attack path by the victim. The victim will not be able to distinguish between marked packet and unmarked packet because the attacker can provide false marking information on the packet header. The victim will have combination of true attack path and forged path and it is hard to identify which one is the true attack path. Park and Lee [13] show that for single-source DoS attacks PPM is effective to localize the origin of attacks. Even if it can not pin down the attacker position in the Internet, the PPM can identify a small set of sources as potential candidates for a DoS attack. In distributed DoS (DDoS) attack, which is not uncommon in the Internet, numerous number of sources are used to attack a victim. Attacker $A1$ can hack machines in different domains and use them, $A2-A5$ in Figure 2, as attacking agents to attack the victim. When the attacker signals the agents, they send packets to the victim V simultaneously. Each source can send small amount of packet to the victim and these will accumulate if large number of agents are used to attack the victim. The traceback is more difficult in this case because each attack agent Ai can send packets that are not enough to identify the source. The attacker can increase the uncertainty in the IP traceback for DDoS attacks [13]. Thus, PPM is vulnerable to distributed DoS attack.

B. Preventing Attacks

We can take preventive solution for DoS attacks caused by IP spoofing. Filtering spoofed packets whenever they are detected is an obvious solution. We discuss several packet filtering techniques in this subsection.

Ingress Filtering. Incoming packets to a network domain can be filtered by a firewall or at ingress routers that perform *customs-type* checking. These entities verify the identity of the packets entering into the domain, like an immigration security system at the airport. Firewalls are effective to stop attacks based on protocol, port, and IP address information. Ingress filtering, proposed by Farguson and Senie [14], is a more rigid and restrictive mechanism to drop traffic with IP address that does not match a domain prefix connected to the ingress router. As an example, in Figure 2, the attacker $A1$ resides in D1=a.b.c.0/24, which is provided Internet connectivity through $R1$. The attacker wants to launch a DoS attack to the victim V that is connected to domain $D4$. If the attacker spoofs IP address of host $H5$ of domain $D5$ having network prefix x.y.z.0/24 for this attack, an input traffic filter on the ingress link of $R1$ thwarts this spoofing. $R1$ only allows traffic originating from source addresses within the a.b.c.0/24 prefix. Thus, the filter prohibits an attacker from using *invalid* source addresses from outside of the prefix range. Ingress filtering can drastically reduce the DoS attack by IP spoofing if *all* domains use it. Similarly, filtering foils the DDoS attack using reflectors. In Figure 2, Ingress filter of

$D2$ will discard packets destined to the reflector $H3$ and specifying $V's$ address in the source address field. Thus, these packets will not be able to reach to the reflector. It is hard to deploy ingress filters in *all* Internet domains. If there are some unchecked points, it is possible to launch DoS attacks from that points. Sometimes legitimate traffic can be discarded by an ingress filtering when Mobile IP is used to attach a mobile node to a foreign network. Packets from a mobile node will have source addresses that do not match with the network prefix where the mobile node is currently attached.

Unlike ingress filtering, egress filtering [15] resides at the exit points of a network domain and checks whether the source address of exiting packets belong to this domain. If not, egress filter will discard the packet to stop attack by spoofed packets. Egress filters do not help to save resource wastage of the domain where the packet is originated but it saves other domains from possible attacks by these packets. Beside the placement issue, both ingress and egress filters have similar behavior.

Route-based Filtering. Park and Lee [16] propose route-based distributed packet filtering. This filter differs from ingress filtering in a sense that, unlike ingress filtering, route-based filters use the route information to filter out spoofed IP packets. For example, in Figure 2, say attacker $A1$ belongs to domain $D1$ is attempting a DoS attack to the target V that belongs to domain $D4$ and uses the spoofed address $H5$ belongs to domain $D5$. The filter at domain $D1$ would recognize that a packet originated from domain $D5$ and destined to target V should not travel through domain $D1$. Then, the filter at $D1$ will discard the packet. The power of route-based filter is that it does not use/store individual host addresses for filtering, rather it uses the topology information of Autonomous System (AS). The authors of [16] show that with partial deployment of route-based filters, about 20% in the Internet AS topologies, it is possible to achieve a synergistic filtering effect that prevents spoofed IP flows reaching other ASes. These filters need to build route information by consulting BGP routers of different ASes. Route on the Internet changes with time [17] and it is a challenge for route-based filters to be updated in real time.

All filters fall short to detect IP address spoofing from the same domain the attacker resides in. For example, in Figure 2, if attacker $A1$ uses some unused IP addresses of domain $D1$, filters can not stop such forged packets to reach the victim V. For for this case, IP traceback is a good candidate to locate the site(s) that could have sent packets and take action accordingly [16].

III. Monitoring to Detect Service Violations and DoS Attacks

An Internet service provider needs to monitor its network domain to ensure the network is operating well. One obvious way of monitoring is to log packets at various points throughout the network and then extract information to discover the path of any packet [18]. This scheme is useful to trace an attack long after the attack has been accomplished. The effectiveness of logging is limited by the huge storage requirements especially for high speed networks. Stone [19] suggested to create a virtual overlay network connecting all edge routers of a provider to reroute *interesting* flows through tunnels to central tracking routers. After examination, suspicious packets are dropped. This approach also requires a great amount of logging capacity.

A QoS network domain needs continuous monitoring of the network for possible service violations and bandwidth theft attacks. Attackers can impersonate a legitimate customer by spoofing flow identities. Network filtering [14] at routers can detect such spoofing if the attacker and the impersonated customer are on different domains, but the attacks proceed unnoticed otherwise. In addition to different traffic classes, a QoS domain has to support best effort (BE) traffic that might lead to SLA violations because edge routers do not have control over BE traffic. The service provider detects any service violation that causes other users to suffer from getting the negotiated QoS. In case of DoS attack, numerous number of flows from different sources are destined to a victim. These flows aggregate on their way as they get closer to the victim. Monitoring can help an upstream network domain to detect these high bandwidth aggregates that could lead to DoS attacks in downstream domains [8], [20].

SLA parameters such as delay, packet loss, and throughput are measured to ensure all users are getting their target share. Delay is an end-to-end latency measurement; packet loss is the ratio of total packets dropped from a flow[1] to the total packets of the same flow entered into the domain; and throughput is the total bandwidth consumed by a flow inside a domain. Delay and loss are important parameters to monitor a network domain because if a network domain is properly provisioned and no user is misbehaving, the flows traversing through the domain should not experience high delay or loss inside that domain. Excessive traffic due to attacks changes the internal characteristics of a network domain. This change of internal characteristics is a key point to monitor a network domain. Bandwidth is used to detect whether any flow is getting more than its share, which causes other flows to suffer. We employ these three parameters to detect SLA violations and DoS attacks. Although, jitter (delay variation) is another important SLA parameter, it is flow-specific and therefore, is not suitable to use in network monitoring. The SLA parameters can be estimated with the involvement of internal (core) routers in a network domain or can be inferred without their help. A large body of research has focused on measuring delay, loss, and throughput in the Internet [21], [22]. The measurement of SLA parameters inside a network domain for monitoring purposes is described later when we explain the monitoring schemes.

We describe two monitoring schemes in this paper to detect SLA violations. One scheme uses the loss inference mechanism using striped unicast probing. We refer to this as *stripe-based* scheme to monitor QoS network domain. We improve on the *stripe-based* monitoring scheme by distributing the probing responsibility over each edge routers in the second scheme, which we call the *distributed* monitoring scheme in this paper. Distributed monitoring scheme has less monitoring overhead and can detect attacks on both directions of a link. Both schemes have an entity, SLA Monitor (SLAM), to collect statistics, analyze them, and decide about violations.

A. Stripe-based Monitoring

To monitor a network domain, SLA components such as delay, loss, and throughput are measured and verified with pre-defined values to detect service violation. This section describes several approaches to measure each SLA parameter and how to use the collected data to detect SLA violations.

Delay Measurements. Delay bound guarantees made by a network provider to customer flows are for the delays experienced by the flows while traversing between ingress and egress edges of the provider's domain. For each packet passing through an ingress router, the ingress copies the packet's IP header into a new packet with a certain pre-configured probability p_{probe}. The ingress encodes the current timestamp $t_{ingress}$ into the payload and marks the protocol field of the IP header with a new value. An egress router recognizes such packets and removes them from the network. Additionally, the egress router computes the delay for a packet of flow i from the difference between the two timestamps. We ignore

[1] flow can be a micro flow with five tuples (addresses, ports, and protocol) or an aggregate one that is combined with several micro flows.

minor drifts of the clocks since all routers are in one administrative domain and can be synchronized fairly accurate. The egress sends a message with the packet details and the measured delay to the monitor, SLAM. At the SLAM, packets are classified as belonging to customer j. Then, the SLAM updates the average packet delay of each customer's traffic as an exponential weighted moving average (EWMA) to put a small weight to past history.

If the average packet delay exceeds the delay guarantee in the SLA, i.e. $avg\_delay^j > SLA^j_{delay}$, we conclude that it is an indication of an SLA violation. If the network is properly provisioned and all flows do not misbehave there should not be any delay greater than SLA^j_{delay} for any customer j. This high delay can be caused by some flows that are violating their SLAs or bypassing the SLA checking, which is an attack. If the delay exceeds a certain threshold, the monitor needs to probe the network for loss. Loss measurement is used to isolate congested links. The congested links are necessary to detect egress and ingress routers involved in high traffic paths, which helps to detect and control DoS attacks respectively.

Loss Measurements. Packet loss guarantees made by a provider network to a customer are for the packet losses experienced by its conforming traffic inside the provider domain. Measuring loss by observing packet drops at all core routers is an easy task. It imposes, however, excessive overhead on the core routers to record each drop entry and periodically send it to the SLA monitor. The monitor can match the destination IP address prefix to detect flows going to a particular network domain for possible DoS attacks. We refer to this scheme as *Core* when we compare all schemes in the next section.

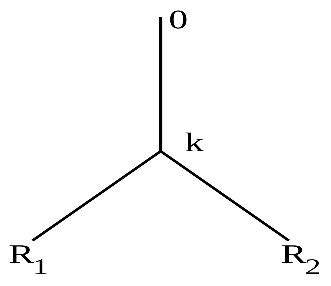


Fig. 3. Binary tree to infer loss from source 0 to receivers R_1 and R_2

The stripe-based probing mechanism [23] is adopted to infer loss characteristics inside a domain without relying on the core routers. This monitoring scheme sends a series of probe packets, called a stripe, with no delay between the transmission of successive packets (usually three packets). We describe how the loss inferring scheme works for unicast traffic as it is described in [23]. Readers are referred to [24] for multicast traffic. For a two-leaf binary tree spanned by nodes 0, k, R_1, R_2, stripes are sent from the root 0 to the leaves to estimate the characteristics of one link, say $k - R_1$ (Figure 3). The first two packets of a 3-packet stripe are sent to R_2 and the last one to R_1. If a packet reaches to any receiver, we can infer that the packet must reached the branch point k. If R_2 gets both packets of a stripe, it is likely that R_1 will receive the last packet of the stripe. The transmission probability, A_k, for node k is expressed in equation (1)

$$A_k = \frac{Z_{R_1} Z_{R_2}}{Z_{R_1 \cup R_2}} \tag{1}$$

where Z_i represents the empirical mean of a binary random variable which takes value 1 when all packets sent to R_i reach their destination and 0 otherwise. The mean is taken over n identical stripes. The loss of $k - R_1$ is inferred using the loss of its parent, k. A complementary stripe is sent similarly to estimate the characteristics of link $k - R_2$. By combining estimates of stripes down each such tree, the characteristics of the common path from $0 - k$ is estimated. This inference technique extends to general trees. Consider an arbitrary tree where for each node k, $R(k)$ denotes the subset of leaves descended from k. Let, $Q(k)$ denote the set of ordered pairs of nodes in $R(k)$ descended from k. For each $(R1, R2) \in Q(k)$, a stripe should be sent from the root to the receivers R_1 and R_2.

The unicast probing scheme is extended in [8] for routers with active queue management, e.g., 3-color RED [25]. This scheme is used to monitor loss inside a QoS network domain. Since, a QoS network has several traffic classes, it is necessary to send probes belonging to different QoS classes. An active queue, such as RED, is configured with different parameters to achieve service differentiation. The probability of acceptance for each class of traffic in the active queue can be calculated using the configuration parameters. Let, $\mathcal{P}_i$, be the percentage of $i \in C$ class of probe packets accepted by the active queue, where C is the set of all traffic classes. Link loss can be inferred by subtracting of transmission probability of equation (1) from 1, i.e. $1 - A_k$. Therefore, if L_i, is the inferred loss for traffic class i, the overall loss is expressed as shown in equation (2), where n_i is number of samples taken from i types of traffic. Only non-zero loss of each class is used to estimate $L_{overall}$. For details, see [8].

$$L_{overall} = \frac{\sum_{i \in C} n_i \times \mathcal{P}_i \times L_i}{\sum_{i \in C} n_i} \tag{2}$$

Throughput Measurements. The objective of checking throughput violation is to ensure nobody is consuming extra bandwidth (beyond the SLA), which starves others. This can not be detected by a single ingress or egress router if the user sends at a lower rate than SLA through multiple ingress routers. To each ingress, it does not violate the SLA but as a whole it does. The service provider may allow a user to take extra bandwidth as long as everybody else is fine. This depends on the policy of the service provider.

The monitor measures throughput by probing globally all egress routers when it sees any violation in delay and loss. Egress routers of a QoS-domain maintain the aggregate rate for each user. This rate is a close approximation of bandwidth consumption by each flow inside the domain [8]. When the monitor gets throughput of all flows from egress routers, it calculates the throughput for user j, as: $B^j = \sum_{i=1}^{N} B^{ij}$, where B^{ij} is bandwidth consumed by user j at edge router i and N is the total number of edges. If $B^j > SLA_{bw}^j$ then it is an SLA violation, otherwise it is not. To detect bandwidth theft that does not increase the internal characteristics of the network, the monitor can periodically poll egress routers.

Violation Detection. When delay, loss, and bandwidth consumption exceed the pre-defined thresholds, the monitor decides on possible SLA violation. The monitor knows the existing traffic classes and the acceptable SLA parameters per class. High delay is an indication of abnormal behavior inside a network domain. If there is any loss for the guaranteed traffic class and if the loss ratios of other traffic classes exceed certain levels, an SLA violation is flagged. This loss can be caused by some flows consuming bandwidths beyond their SLA_{bw}. Bandwidth theft is checked by comparing the total bandwidth achieved by a user against the user's SLA_{bw}. The misbehaving flows are controlled at the ingress routers.

To detect DoS attacks, set of links L with high loss are identified. For each congested link, $l(v_i, v_j) \in L$, the tree is pruned into two subtrees. One subtree has egress routers as leaves through which high aggregate bandwidth flows are leaving. This subtree is obtained from leaves descendant from v_j. If many exiting flows have the same destination IP prefix, we can decide that either this could be a DoS attack or they are going to a popular site [20]. Decision can be taken with consulting the destination entity. If it is an attack, we can stop this by triggering filters at the ingress routers that are leaves of the *other* subtree descendant from v_i and feeding flows to the congested link. For each violation, the monitor takes action such as throttling a particular user's traffic using a flow control mechanism.

B. Distributed Monitoring

The single-point probing of *stripe-based* scheme can be improved by distributing the probing agents over all edge routers. We propose the distributed monitoring approach to reduce the monitoring overhead to detect service violation and DoS attacks. This mechanism uses an overlay network infrastructure. Peers (edge routers) of an overlay network know each other by forming a virtual network on top of the physical network. Figure 4(a) shows the spanning tree of a network topology. The edge routers of this network domain form an overlay network among themselves, Figure 4(b).

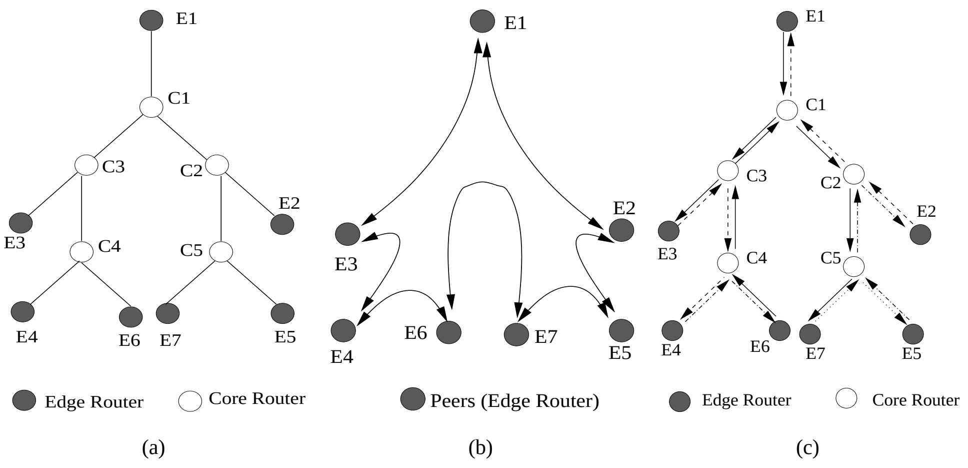


Fig. 4. Network monitoring using the distributed mechanism. (a) Spanning tree of a network domain (b) Overlay architecture, every node has a virtual link with each of its neighbors (c) Internal links' direction for each probing.

In the distributed monitoring approach, each edge router knows its right and left neighbors and can probe part of the network. The SLA Monitor (SLAM) sits at the root of the spanning tree. The SLAM probes the network regularly for unusual delay patterns inside the network. The delay and throughput measurements are same as described in *stripe-based* scheme. The two schemes differ in measuring loss. Service violation can be detected without exact loss values, rather it requires whether a link has higher loss than specified threshold or not. The link with high loss is referred to as a congested link. The goal of distributed probing is to detect all congested links.

When delay goes high, the SLAM triggers agents at different edge routers for loss probing. Each edge router probes its right neighbor first. In Figure 4(b), edge router $E1$ probes the link $E1 - E3$, $E3$ probes the link $E3 - E4$, $E4$ probes the link $E4 - E6$, $E6$ probes the link $E6 - E7$, and so on. Let, X_ρ be a boolean random variable that represents the output of probe $\rho \in \mathbf{P}$, where $\mathbf{P}$ is the set of probes needed. X_ρ takes on value 1 if the measured loss exceeds the threshold in any link throughout the probe path and 0 otherwise. For example, if the outcome of probing $E1 - E3$ path is 1, it means either $E1 - C1$, $C1 - C3$, $C3 - E3$, or a combination of them is congested. The internal links of each probe are shown in Figure 4(c). If the outcome is 0, definitely all of the internal links are *not* congested. In this way, we write equations to express all internal links in terms of probe outcomes. The set of edge routers that will perform probing, left and right neighbors of each edge router, and the equations to express the congestion status of internal links in terms of outcome of each probe can be obtained by traversing the tree using Depth First Search from the root.

We note that loss in path $E1 - E3$ might not be same as loss in path $E3 - E1$. This path asymmetry phenomenon is shown in [26]. The distributed monitoring can detect high loss in both directions of a link. To solve the set of equations in the first round of probing, we need another round of probing in the clock-wise direction from $E1$ to $E2$, $E2$ to $E5$, and so on. These two sets of equations are enough to isolate congested links with close approximation. The distributed monitoring can detect congestion on any non-overlapping probe path in any direction. The path may have arbitrary number of internal links. This is because if we have congestion in one path, we have sufficient probes to isolate each internal link for this probe path. As boolean equations may not have unique solutions in some cases, distributed monitoring can localize congested links on overlapping paths with close approximation.

We give a brief description of the algorithm for detecting congested links. (The details are not given here due to space limitation.) If the outcome of any probe is zero (loss does not exceed the threshold), a zero value is assigned to all internal links that comprise the probe path on the probing direction. At the end of probing, if each internal link is assigned either 1 or 0 values, we are done. But sometimes, we won't be that lucky. Having congestion on links that affect multiple probe paths in same direction will eventually lead to some boolean equations that do not have unique solutions. In that case, we can use some special set of probes that were not used before or apply *stripe-based* probing only to a part of the tree to determine the exact locations of the congested links. We can even report all the internal links as congested from the unsolved equations, which is a close approximation of the actual results. Because this situation is possible only when majority of the internal links among the total links in the equation set are congested in a probe path. After identifying links with high loss, we prune the whole tree T to isolate ingress and egress routers to detect high bandwidth aggregates and detect service violation and DoS attacks as it is described in *violation detection* of stripe-based scheme. The distributed monitoring scheme can be extended for QoS networks using probes with different classes similar to what is described in stripe-based method.

Advantages. The advantages of using *distributed* monitoring are listed as below:

1. The distributed scheme requires less number of total probes, $O(n)$, to estimate links with high loss than stripe-based scheme, which requires $O(n^2)$ where n is number of edge routers in the domain.
2. The distributed scheme is able to detect violation in both directions of any link in the domain, whereas the stripe-based can detect any violation only if the flow direction of misbehaving traffic is same as the probing direction from the root. To achieve same ability like distributed one, the stripe-based needs to probe the whole tree from several different points, which will increase the injected traffic in the domain.
3. The distributed scheme can use TCP based loss measurement to detect loss of both directions in one probe cycle (e.g. *Sting* [26]).
4. In stripe based scheme, two leaves/receivers are probed at a time. It takes long time to complete probing the whole tree. If all leaves are probed simultaneously, $E1 - C1$ link will face huge amount of traffic at that time. On the other hand,

the distributed scheme can do parallel probing naturally.

IV. Comparison of Different Schemes

In this section, we conduct a quantitative analysis of the overhead imposed by each scheme used to detect/prevent DoS attacks. The schemes we compare here are: Ingress Filtering (*Ingf*), route-based packet filtering (*Route*), traceback with probabilistic packet marking (*PPM*), core-assisted scheme that observes packet drops at core routers and sends drop history to the monitor to detect attacks (*Core*), stripe-based monitoring (*Stripe*), and distributed monitoring using overlay network architecture (*Distributed*). We account for the communication overhead due to packets injected into a network for probing as well as the computation overhead due to extra processing at routers. The communication overhead is computed as the number of extra *bytes* (not packets) injected per unit time. For computation overhead, the extra processing at routers could contain: more address lookups, changing some header fields, checksum re-computation, and any CPU processing needed by the scheme. For simplicity, we charge the schemes only one processing unit (α) per packet that needs one or more of the aforementioned processing. We consider a domain $\mathcal{D}$ with M edge routers and N core routers. We assume there are F flows traversing through each edge router and each flow has P packets on average. We define θ as the percentage of misbehaving flows that cause DoS attacks.

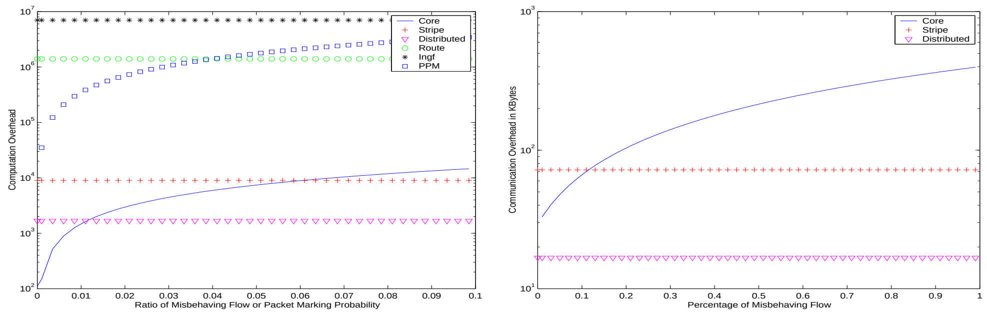
Filtering techniques do not incur any communication overhead but they need one extra address lookup (for source address) per packet. Ingress filtering has a total computation overhead of $M \times F \times P \times \alpha$ per unit time because it checks the source address of every packet. We need to deploy ingress filters in every domain in the Internet to effectively stop all possible attack. The route-based filtering scheme, on the other hand, does not require every single domain to have a filter. Park, et al showed that placing this filter at approximately 20% of all autonomous systems can prevent DoS to a great extent [16]. For a domain that deploys a router-based filter, the overhead is the same as the ingress filter. Globally speaking, the overhead of route-based filtering is one fifth of the overhead of ingress filtering on the average. In our comparison, we use $Overhead_{Route} = \frac{1}{5}Overhead_{Ingf}$.

The *PPM* does not incur any communication overhead but adds one extra processing unit (α) for every packet that gets marked at all intermediary routers. Traceback with *PPM* needs to mark packets with probability p at each router on the path to the victim. If a packet passes through h hops, on the average, in the network domain $\mathcal{D}$, the computation overhead is $M \times F \times P \times p \times h \times \alpha$ per unit time.

The monitoring schemes inject probe traffic into the network and add computation overheads as well. The total injected probes and size of each probing packet are used to calculate the communication overheads in terms of bytes. The *Core* scheme depends on the number of packets core routers send to the monitor to report drop history. The drop history at each core router depends on the flows traversing the network domain and the percentage of these flows that are violating their SLAs at a particular time. For the domain $\mathcal{D}$, if d bytes are required to record drop information of each flow, then each core needs to send $C = max(1, \frac{F \times \theta \times d}{packet\_size})$ control packets to the monitor. To obtain *loss ratio,* the monitor queries all edges for packet count information of the misbehaving flows. Every edge will reply to this query. The total number of packets exchanged is $(2M + N) \times C$ packets. Therefore, the communication overhead is $(2M + N) \times C \times packet\_size$. The $packet\_size$ is a configurable parameter. The computation overhead is $(2M + N) \times C \times h \times \alpha$, where h is the average number of hops a packet traverses.

In the stripe-based monitoring scheme, a stripe of s packets is sent from the monitor to every egress routers pair. For the network domain $\mathcal{D}$, the total number of probing packets is $s \times (M-1) \times (M-2) \times f$, where f is the frequency by which we need to send stripes per unit time. The communication overhead is $s \times (M-1) \times (M-2) \times f \times packet\_size$. The computation overhead is $s \times (M-1) \times (M-2) \times f \times \alpha \times h$, where h is the average number of hops a packet traverses.

For the distributed monitoring, each edge router probes its left and right neighbors. If it requires f' probes per unit time, the communication overhead is $2 \times M \times f' \times packet\_size$ per unit time. The computation overhead is $2 \times M \times f' \times \alpha$.



(a) Computation overhead

(b) Communication overhead

Fig. 5. Overhead per unit time for the network domain shown in Figure 4 with parameters F=100,000 flows, s=3 packets, f=20, f'=30, and average packets per flow is 10. Filtering and PPM does not have communication overhead

To visualize the differences among all schemes, we plot the computation and communication overhead for the domain shown in Figure 4 with the following parameters: number of edge routers M is 7, number of core routers N is 5, the stripe length s is 3 packets, number of flows F is 100,000, frequency of probing f is 20 and f' is 30. Figure 5(a) shows that filters and PPM have much higher computation overheads than monitoring schemes. Note that X-axis carries two parameters: the marking probability, which is used for PPM scheme and the ratio of misbehaving flows for the rest of the schemes. The gain of the *Distributed* scheme over the *Stripe* scheme is not significant in this example but the difference will be much higher when the number of edge routers is high. Figure 5(b) compares the communication overhead of all schemes except filtering and *PPM* because they do not have any communication overhead. It shows probes injected by *Stripe* consumes less than 100K bytes of bandwidth per unit time—less than 0.2% of the capacity of an OC3 link. The *Distributed* scheme consumes less than the *Stripe* one. The communication overhead of the *Distributed* and *Stripe* schemes are lower than that of the *Core* scheme. This is because the control packet size of *Core* is larger than the probe packet size. *Core* uses control packet size equal to the maximum transmission unit of the network to minimize total packets needs to be sent whereas the probe packet size is 20 bytes with 20 bytes of IP header.

We summarize some important features of all schemes in Table I. Ingress filtering and core-assisted monitoring have high implementation overhead because the former needs to deploy filters at all ingress routers in the Internet and the latter needs support from all edge and core routers in a domain. All monitoring schemes need clock synchronization, which is an extra overhead but they can detect service violations as well. Filters are proactive in nature and all other schemes are reactive. Filters can detect attacks by spoofed packets whereas rest of the schemes can detect attack even the attacker does

Property	PPM	Ingress Filtering	Route-based	Core-assisted	Stripe	Distributed
Overhead depends on	attack volume	number of incoming packets	number of incoming packets	number of flows violating SLAs	routers, topology, attack traffic	routers, topology, attack traffic
Implementation overhead	all routers	all ingress edge routers	all routers of selective domains	all edge and core routers	all edge routers	all edge routers
Clock synchronization	—	—	—	at edge and core routers	at edge routers	at edge routers
Response	reactive	proactive	proactive	reactive	reactive	reactive
SLA violation detection	no	no	no	yes	yes	yes
Detect attacks initiated using	any IP	spoofed IP from other domains	spoofed IP from other domains	any IP	any IP	any IP

TABLE I

COMPARISON AMONG DIFFERENT SCHEMES TO DETECT/PREVENT SERVICE VIOLATIONS AND DOS ATTACKS

not use spoofed IP addresses from other domains.

V. SUMMARY

We have investigated several methods to detect service level agreement violations and DoS attacks. Both preventive and reactive methods are discussed and compared among each other. IP traceback is an efficient method to locate the source of an attack with close proximity by probabilistic marking packets at routers. This is used after realizing that an attack has happened. Hash-based IP traceback provides a source path isolation engine (SPIE) to identify the source of a particular IP packet. Thus, SPIE can detect network path of an attack even for low volume of packets received by the victim. Ingress filtering provides safety against IP spoofing by checking the source address of a packet. The route based packet filtering uses network topology information to filter out spoofed packets. Strategic placing of route based filters make the deployment attainable. Both filtering approaches are preventive in nature and can be used along with a traceback mechanism. When filters fail in detecting an attack, traceback provides ways to locate and possibly punish an attacker. We showed that bandwidth theft attacks are likely to happen in a QoS network and network monitoring can detect service violations as well as DoS attacks. We briefly described stripe-based and distributed network monitoring schemes in this paper. The on-demand probing of the monitoring schemes reduce the extra traffic injected by the probes. Both monitoring approaches can be integrated with an admission control scheme to dynamically regulate traffic and stop an attack as soon as it is detected. Furthermore, the monitoring techniques can be used in any general network architecture (not only a QoS network).

Acknowledgments

This research is sponsored in part by the National Science Foundation grants CCR-001712 and CCR-001788, CERIAS, an IBM SUR grant. The authors thank Abdul M. Khan for his valuable comments on the distributed monitoring approach.

References

[1] D. F. DeLong, "Hackers said to cost U.S. billions," *E-Commerce Times Online*, Feb 8, 2001.

[2] D. Moore, G. M. Voelker, and S. Savage, "Inferring Internet denial-of-service activity," *in Proc. USENIX*, 2001.

[3] L. Garber, "Denial of Service attacks rip the Internet," *IEEE Computer* , vol. 33,4, pp. 12–17, April 2000.

[4] G. Spafford and S. Garfinkel, *Practical Unix and Internet Security*, O'Reilly & Associates, Inc, second edition, 1996.

[5] S. Savage, D. Wetherall, A. Karlin, and T. Anderson, "Network support for IP traceback," *IEEE/ACM Transaction on Networking*, vol. 9:(3), pp. 226–237, June 2001.

[6] C. Barros, "A proposal for ICMP traceback messages," Internet Draft http://www.research.att.com/lists/ietf-itrace/2000/09/msg00044.html, Sept 18, 2000.

[7] V. Paxson, "An analysis of using reflectors for distributed denial-of-service attacks," *ACM Computer Communication Review*, vol. 31 (3), July 2001.

[8] A. Habib, S. Fahmy, S. R. Avasarala, V. Prabhakar, and Bharat Bhargava, "On detecting service violations and bandwidth theft in QoS network domains," Tech. Rep., CSD-01-22, Department of Computer Sciences, Purdue University, Dec 2001.

[9] H. Burch and H. Cheswick, "Tracing anonymous packets to their approximate source," *in Proc. USENIX Conference.*, pp. 319–327, Dec. 2000.

[10] D. Song and A. Perrig, "advanced and authenticated marking schemes for IP traceback," *in Proc. IEEE INFOCOM*, Apr. 2001.

[11] D. Dean, M. Franklin, and A. Stubblefield, "An algeraic approach to IP traceback," *in Proc. Network and Distributed System Security Symposium*, Feb. 2001.

[12] A Snoeren, C. Partridge, L. Sanchez, W. Strayer, C. Jones, and F. Tchakountio, "Hashed-based IP traceback," *ACM SIGCOMM*, Aug. 2001.

[13] K. Park and H. Lee, "On the effectiveness of probabilistic packet marking for IP traceback under Denial of Service attack," *in Proc. IEEE INFOCOM*, Apr. 2001.

[14] P. Ferguson and D. Senie, "Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing agreements performance monitoring," RFC 2827, May 2000.

[15] SANS Institute, "Egress filtering v 0.2," http://www.sans.org/y2k/egress.htm, Feb 2000.

[16] K. Park and H. Lee, "A proactive approach to distributed DoS attack prevention using route-based packet filtering," *in Proc. ACM SIGCOMM*, Aug 2001.

[17] V. Paxson, "End-to-end internet packet dynamics," *n Proc. SIGCOMM '97*, 1997.

[18] G. Sager, "Security fun with OCxmon and cflowd," Intenet2 working group meeting, Nov 98.

[19] R. Stone, "Centertrack: An IP overlay network for tracking DoS floods," *In Proc. USENIX Security Symposium*, Aug 2000.

[20] M Mahajan and et al, "Controlling high bandwidth aggregates in the network," Technical Report, ACIRI, Feb 2001.

[21] V. Paxson, *Measurement and Analysis of End-to-End Internet Dynamics*, Ph.D. thesis, University of California, Berkeley, Computer Science Division, 1997.

[22] V. Paxson, G. Almes, J. Mahdavi, and M. Mathis, "Framework for IP performance metrics," IETF RFC 2330, May 1998.

[23] N. G. Duffield, F. Lo Presti, V. Paxson, and D. Towsley, "Inferring link loss using striped unicast probes," *in Proc. IEEE INFOCOM*, April Alaska, April 2001.

[24] R. Cáceres, N. G. Duffield, J. Horowitz, and D. Towsley, "Multicast-based inference of network-internal loss characteristics," *IEEE Transactions on Information Theory*, Nov 1999.

[25] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, August 1993, ftp://ftp.ee.lbl.gov/papers/early.ps.gz.

[26] S. Savage, "Sting: a TCP-based network measurement tool," *in Proc. USENIX Symposium on Internet Technologies and Systems (USITS '99)*, Oct 1999.