SOS: Secure Overlay Services

Angelos D. Keromytis[*] Vishal Misra[*†] Dan Rubenstein[†*]
[*]Department of Computer Science [†]Department of Electrical Engineering
Columbia University
New York, NY
{*angelos,misra,danr*}@cs.columbia.edu

Columbia University Technical Report EE200415-1
February, 2002

Abstract

Denial of service (DoS) attacks continue to threaten the reliability of networking systems. Previous approaches to protect networks from DoS attacks are *reactive* in that they wait for an attack to be launched before taking appropriate measures to protect the network. This leaves the door open for other attacks that use more sophisticated methods to mask their traffic. We propose an architecture called *Secure Overlay Services (SOS)* that *proactively* prevents DoS attacks, geared towards supporting Emergency Services or similar communications. The architecture is constructed from a combination of secure overlay tunneling, routing via consistent hashing, and filtering. We reduce the probability of successful attacks by (i) performing intensive filtering near protected network edges, pushing the attack point perimeter into the core of the network where high-speed routers can handle the volume of attack traffic and (ii) introducing randomness and anonymity into the architecture, making it difficult for an attacker to target nodes along the path to a specific SOS-protected destination. Using simple analytical models, we evaluate the likelihood that an attacker can successfully launch a DoS attack against an SOS-protected network. Our analysis demonstrates that such an architecture reduces the likelihood of a successful attack to minuscule levels.

1 Introduction

In the immediate aftermath of the September 11 events in New York City, the Internet was used to facilitate communication between family members, as the phone network was overwhelmed. It does not require a great leap of faith to imagine using the Internet as a communication medium for crisis and emergency response teams, especially when using a wireless network substrate. In particular, the network would be used to protect communications between widely dispersed "static" sites (*e.g.,* various federal, state, and city agencies) and (semi-) roaming stations and users. The communication path between the various sites and the emergency response teams (ERTs) needs to be kept clear of interference; a denial of service (DoS) attack on such a network could seriously disrupt the rescue and recovery effort, at minimal cost and danger to the attacker.

A proposal to build a completely separate network is currently under consideration by the US government (appropriately called "GovNet"). Such a network would have a high cost and would likely fall behind the general-purpose Internet as new technologies are discovered and deployed. The network security community is also doubtful of the claims of increased security that such a separate network would entail. It is therefore a worthwhile endeavor to consider building a secure infrastructure either inside or upon the existing Internet. Building such a service raises several questions: How should priorities be assigned during an emergency? How much modification is required to the underlying infrastructure? How can communications be secured?

In this paper, we address the problem of securing a communication service on top of the existing IP infrastructure from DoS attacks. We assume that there is a subset of clients scattered throughout the wide-area network who require access to a particular destination that should otherwise be secured. An example is a database that maintains timely or confidential information such as building structure reports, intelligence, or strategic information. Users in the field (emergency workers, government agents, police, *etc.*) should be able to access this information from any location (*i.e.,* any IP address) within the wide area network, since it is not always possible to predict their locations when emergencies strike. We also assume that there is a set of users that want to prevent access to this information, and will launch DoS attacks upon any network points to help them achieve this goal. The goal of the attackers is to identify any "pinch" points in the communications substrate and render them inoperable by flooding them with large volumes of traffic. An example of an obvious attack point is the location (IP address) of the destination that is to be secured, or the routers in its immediate network vicinity.

Previous approaches that address this problem are *reactive*: they monitor traffic at a target location, waiting for an attack to occur. After the attack is identified, typically via analysis of traffic patterns and packet headers, filters are established in an attempt to block the offenders. The main two problems with this approach are the accuracy with which legitimate traffic can be distinguished from the DoS traffic, and the robustness of the mechanism for establishing filters deep enough in the network (away from the target) that the effects of the attack are minimized.

Our approach is to be *proactive*. In a nutshell, the portion of the network immediately surrounding the target (location to be protected) aggressively filters and blocks all incoming packets whose source addresses are not "approved". The small set of source addresses that are "approved" are kept secret so that attackers cannot use them to patch through the filter. A distributed set of nodes throughout the wide area network form a *secure overlay*: any transmissions that wish to traverse the overlay must first be validated at entry points of the overlay. Once inside the overlay, the traffic is tunneled securely for several hops along the overlay to the "approved" and secret locations which can then forward the validated traffic through the filtering routers to the target. The main three principles behind our design are

- elimination of communication "pinch" points, which constitute attractive DoS targets.
- the ability to recover from random or induced failures of the forwarding infrastructure or secure overlay nodes.
- utilization of high-capacity nodes with good connectivity, which are less affected by a DoS attack than nodes in the periphery of the network.

This paper proposes a preliminary approach to constructing this forwarding service that we refer to as a *Secure Overlay Service,* or *SOS* for short. We discuss how to design the architecture of the overlay that is secure with high probability, given attackers that have a large but finite set of resources to perform the attacks, as well as tthe following information:

- the IP addresses of the nodes that participate in the overlay and of the target that is to be protected.
- the details of the operation of protocols used to perform the forwarding.

Our architecture leverages heavily off of previous work on IP security[KA98, BIK01], IP router filtering capabilities, and novel approaches to routing in overlay[ABKM01] and peer-to-peer (P2P) networks[SMK$^+$01, DKM$^+$01]. We perform a preliminary stochastic analysis using simple networking models to evaluate the likelihood that an attacker is able to prevent communications to a particular target. We determine this likelihood as a function of the aggregate bandwidth an attacker is able to utilize in a DoS attack by exploiting compromised systems. Our analysis includes an examination of the capabilities of static attackers who focus all their attack resources on a fixed set of nodes, as well as attackers who adjust their attacks to "chase after" the repairs that the SOS system implements when it detects an attack. We show that even attackers that are able to launch massive attacks are very unlikely to prevent successful communication. For instance, attackers that are able to launch attacks upon 50% of the nodes in the overlay have roughly one chance in one thousand of stopping a given communication from a client whose connection has minimal access to overlay nodes.

The remaining of this paper is organized as follows: Section 2 overviews related work. In Section 3 we describe the SOS architecture, while Section 4 provides an analysis of the expected behavior of an SOS network under attack. Section 5 shows how an SOS-based architecture can be constructed using off-the-shelf components and protocols, and the next section, Section 6, discusses our plans for future work. Finally, Section 7 concludes the paper.

2 Related Work

One fundamental design principle of the IP architecture is that communication properties that must hold end-to-end should be provided by mechanisms at the end points. This principle, commonly referred to as the "end-to-end principle"[SRC84, Cla88], has been the basic premise behind protocol design. However, as has been demonstrated in the past few years [Tea96, SKK$^+$97, HB96], such mechanisms are inadequate in addressing the problem of DoS attacks: attacks that attempt to overwhelm the processing or link capacity of the target site (or routers that are topologically close) by saturating it with bogus packets.

It is trivial to abuse[SCWA99] or simply ignore congestion control mechanisms, and there are plenty of protocols that have no provision for congestion control. Furthermore, no great technical sophistication is required in launching one of these attacks. Even relatively large-scale DoS attacks (Distributed DoS — DDoS)[1] are not very difficult to launch, given the lack of security in certain email clients and the ability to cause arbitrary code to be executed by an email recipient.

Unfortunately, as a result of its increased popularity and usefulness, the Internet contains both "interesting" targets and enough malicious (or simply ignorant) users that DoS attacks are simply not going to disappear on their own; indeed, although the press has stopped reporting such incidents, recent studies have shown a surprisingly high number of DoS attacks occurring around the clock throughout the Internet [MVS01]. Worse, the Internet is being used for increasingly time-critical applications (*e.g.,* electricity production monitoring and coordination between different generators).

The need to protect against or mitigate the effects of DoS attacks has been recognized by both the commercial and the research world, and some work has been done towards achieving these goals, *e.g.,* [IB02, DFS01, SWKA01, SWKA00]. These mechanisms focus on detecting the source of DoS attacks in progress and then countering them, typically by "pushing" some filtering rules on routers as far away from the target of the attack (and close to the source) as possible. Thus far, the focus of DoS-countering mechanisms has been on *reaction*. The motivation behind this approach has been twofold: first, it is conceptually simple to introduce a protocol that will be used by a relatively small subset of the nodes on the Internet (*i.e.,* ISP routers), as opposed to requiring the introduction of new protocols that must be deployed to and used by end systems. Second, these mechanisms are fairly transparent to protocols, applications, and legitimate users.

Unfortunately, simple detection of the source of a DoS attack is not by itself particularly useful:

[1]For the remainder of this paper we will use the term "DoS" to mean both single host and distributed DoS attacks.

- Since the Internet spans multiple administrative domains and (legal) jurisdictions, it is often very difficult (if not outright impossible) to shut down an attack by contacting the administrator or the authorities closest to the source. In any case, such action cannot be realistically delivered in a timely fashion (*e.g.,* in the order of at most a few hours).
- Even if this were possible, it is often the case that the source of the attack is not the real culprit but simply a node that has been remotely subverted by a cracker. The attacker can simply start using another compromised node.
- Using a "pushback"-like mechanism[IB02] in trying to counter a DoS attack makes close cooperation among different service providers necessary: since most attacks use random source IP addresses (and since ingress filtering is not widely used), the only reliable packet field that can be used for filtering is the destination IP address (of the target). If filters can only be pushed "halfway" through the network between the target and the sources of the attack, the target runs the danger of voluntarily cutting off all of its communications with the rest of the Internet. The accuracy of such filtering mechanisms improves dramatically the closer it is possible to "push" filters to the actual source(s) of the attack. Thus, it is necessary for providers to allow others (providers, or even end-network administrators) to install filters on their routers. Quite apart from the possibility of abuse, it is questionable whether such collaboration can easily be achieved to the degree necessary.

Thus, a different approach is needed in protecting the communications of parties involved in a critical task from the effects of DoS attacks.

3 Architecture Description

The goal of the SOS architecture is to allow communication between a *confirmed source point* and a *target*. By confirmed, we mean that the target has given prior permission to a specific user that currently resides at the source point. Typically, this means that the source must be authenticated and authorized by the SOS infrastructure before traffic is allowed to flow between itself and the target through the overlay. We shall see in Section 5 how this can be efficiently achieved for a large collection of SOS nodes and users.

Furthermore, we assume that there exist attackers in the network that are interested in preventing traffic from reaching the target. These attackers have the ability to launch DoS attacks from a variety of points around the wide area network that we call *compromised locations*. The number and bandwidth capabilities of these compromised locations determine the intensity with which the attacker can bombard a node with packets, effectively shutting down that node's ability to process legitimate traffic. Without an SOS, knowledge of the target's IP address is all that is needed in order for a moderately-provisioned attacker to bring down the target site.

Last, we assume that the set of nodes that participate in the SOS are known to the public. In effect, no node's identity is kept hidden. However, certain jobs that a node may perform in the delivery of traffic are kept secret from the public.

Figure 1 gives a high-level overview of the SOS architecture that protects a target node or site so that it only receives valid transmissions. The various components of the SOS architecture (discussed in more detail later in this section) are:

- **Targets**: Target nodes wish to receive transmissions from validated sources and wish to be protected from phony (*i.e.,* un-authenticated) transmissions. Heavy filtering is applied in the immediate vicinity of the target to protect it from unwanted traffic. We assume that attackers do not have access to routers inside the filtered region (*i.e.,* they cannot observe which source addresses can proceed through the

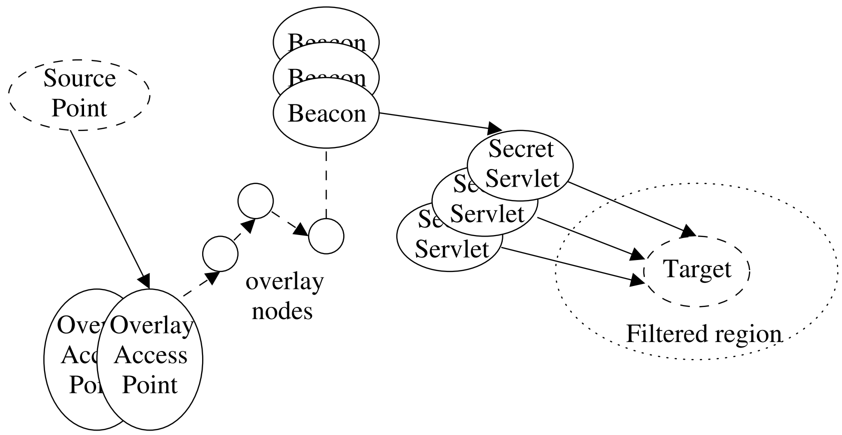


Figure 1: Basic SOS architecture.

filter). Past history indicates that it is a lot more difficult for an attacker to completely take over a router or link in the middle of an ISP's network than to attack an end-host; intuitively, this is what we would expect, given the limited set of services offered by a router (compared to, *e.g.,* a web server or a desktop computer).

- **Secret Servlets**: Nodes that participate on the overlay and act as the (only) entry point to a target. Their identities are kept as secret as possible.
- **Beacons**: A beacon is a node that participates on the overlay. It receives traffic destined for a particular target and, after verifying the legitimacy of the traffic, forwards it to a secret servlet. Hence, beacons are aware of the identities of some of the secret servlets for the targets for which they act as a beacon.
- **Overlay Access Point (OAP)**: A node that participates on the overlay that accepts traffic from "approved" source points that wish to use the overlay to reach a given destination.
- **Source points**: A node on or off the overlay that wishes to send a (legitimate) transmission to a target. It is assumed that source points have been granted permission by the target during an earlier exchange (*e.g.,* have received an appropriate certificate through e-mail).
- **Attack point**: Any node that has been compromised and can be used to launch an attack or snoop the source from where a packet came or destination to where a packet is going (both next hop and final).

Neither the source point, target points, or attack points are assumed to be members of the overlay. An overlay routing protocol is used to deliver packets received at an overlay access point to a beacon. We now discuss the details of the design of this architecture in greater detail.

3.1 Filtering at the target

To protect a target from DoS attacks, a filtering mechanism must be deployed at nodes such that filtering is performed along all paths that lead to the target. We assume that filtering is done at a set of high-powered routers such that i) these routers can handle high loads of traffic, making them difficult to attack, and ii) possibly there are several, disjoint paths leading to the target, each of which is filtered independently. This way, if one of these paths is brought down, filtered traffic still can traverse the others.

Filtering offers an interesting tradeoff in IP networks. By increasing the fraction of IP addresses that are suppressed by the filter, one decreases the potential power of DoS attacks since attackers must guess or somehow infer the source addresses that are not filtered to mount a meaningful attack. However, one also decreases the set of locations from which legitimate transmissions are allowed to originate. When using network overlays, it is still possible for a packet of arbitrary origin to reach the target even when intensive filtering is applied (without spoofing). This is accomplished by forwarding traffic to locations in the overlay whose addresses are permitted to pass through the filter. These locations then forward traffic through the filter. To provide security, two properties must hold:

- Attackers should not be given the identities of the IP addresses of the nodes that can proceed through the filter. Otherwise, an attacker could pass through the filter by simply spoofing the IP address.
- Legitimate clients at confirmed source points should be able to reach the nodes with unfiltered IP addresses.

Thus, the problem of protecting the target from DoS attacks has been converted to two (easier) problems: a) keeping the identities of non-filtered addresses secret while b) allowing traffic from confirmed sources to reach those non-filtered addresses.

3.2 Secret Servlets

We say that a node N_s is a *secret servlet* for a target node N_t if the filter around N_t permits packets whose source address is (the IP address of) N_s to pass through the filter. The set of secret servlets used by a given target N_t is selected by the target itself. This is not difficult, since we assume that the list of nodes that participate in the overlay is available to the public. The target notifies these nodes (in private) of their role as secret servlets. If, for some reason, the target wishes to alter the set of nodes that act as secret servlets, it simply contacts the set of new nodes to inform them of their new role, contacts the set of old nodes to inform them that they will no longer function as secret servlets, and accordingly adjusts the filtering mechanism that protects it. Notice that this process is much simpler than the more general filter-push mechanisms that have been proposed as ways of countering DoS attacks: the set of routers that the target needs to notify is fixed (and thus long-term arrangements can be made with the operators of these routers), and the types of filtering rules that need to be established are fairly straightforward (and can thus be easily verified by these routers without human intervention or complicated resolution mechanisms).

As long as legitimate traffic can find its way onto the SOS overlay, and the SOS overlay is able to direct this traffic to the appropriate secret servlet, the traffic can proceed through to the target destination.

3.3 Access Points

An overlay will be used to tunnel traffic from a legitimate client at a confirmed source point to a secret servlet which can then pass the traffic through the filter to the target. However, the legitimate client might not reside at a node which is part of the overlay. The first step is therefore to give the client access to the overlay. A subset of nodes that participate in the overlay act as *access points* for legitimate clients. The IP addresses of access points may be made public, or may only be revealed to legitimate clients. The security of our architecture does not depend on these IP addresses being secret.

A legitimate client chooses a node N_A from a list of access points and initiates a secure communication with that node using a protocol such as IPsec[KA98]. Hence, when N_A agrees to act as the access point for this client, it has confirmed both the client's right to communicate with the target as well as the IP address of the client. Subsequent traffic between the access point and the client may be protected (again, by IPsec). As an alternative, the client may be given a "cookie" that it has to include into any subsequent packets it sends to the access point. The choice between the two mechanisms depends largely on the perceived threat model: if attackers cannot eavesdrop the communication path between access points and sources, then the cookie approach is sufficient. In a more hostile environment (*e.g.,* when using a wireless network to connect to the Internet), a cryptography-based mechanism may be used.

For additional protection, the cookie or the IPsec state (called "Security Association", or SA) can expire after a certain amount of use (either timed or in terms of the number of packets transmitted). If N_A fails for any reason (including a DoS attack upon N_A), the legitimate client can simply move to another access point elsewhere in the network to continue transmitting to the target. Thus, a DoS attack that accidentally (or purposely) hits the access point or a secret servlet will not permanently affect communications between the source and the target, as both contact points to the SOS overlay can be changed.

So far, we have discussed a mechanism that restricts entry to the overlay to legitimate traffic from confirmed source points that can be classified per flow, as well as a mechanism that permits traffic to go from specific nodes on the overlay to a target through heavy filtering. What is still needed is a means to route through the overlay in a manner that makes it difficult to attack the route.

3.4 Overlay Routing

Routing used within SOS must be robust to attacks upon nodes within the overlay. In particular, if a given set of overlay nodes is attacked, there must be an efficient transition to an alternative path where no nodes are under attack. Our approach is to apply the consistent hashing approach used within the Chord service [SMK+01]. For our purposes, Chord can be viewed as a routing service with the following properties:

- The service is implementable atop the existing IP network structure.
- Given a key that relates to some piece of information (usually a file or piece of content), Chord provides a means to map (hash) the key to a particular subset of nodes that i) are active members of the overlay and ii) contain the information that is associated with the key. Chord also provides an efficient mechanism that routes packets toward one of those members (using $O(\log N)$ nodes in the overlay).
- It is simple to produce multiple mappings (hash functions) that produce different paths to different sets of destination nodes (*i.e.,* each path can be thought of as being selected at random).
- The service is robust to changes in overlay membership.
- Not all nodes that route a packet within Chord using key k need to know the IP address of the final destination to which Chord routes the packet. They only need to know that they are sending the packet to a node on the overlay that is in the "right direction". In this manner, the Chord service assures that a destination exists that is expecting to be the destination for packets with key k.

Any node that is a destination of a route using a key formed by hashing upon the target's IP address is called that target's *beacon* for that hash function. When a packet is approved by an access point for transmission, the hash on the IP address of the target is used as the key. Hence, Chord provides a robust and reliable while relatively unpredictable means of routing packets from an access point to one of several beacons. The final step in the architecture involves getting packets from beacons to secret servlets.

There are several approaches for accomplishing this final step of connecting a beacon to a secret servlet. Here, we describe the simplest. Since any node in the Chord overlay can route to a beacon, it follows that a secret servlet can also contact a beacon for a particular target, for any hash function applied. We require that nodes that act as beacons respond to queries (transmitted securely over the overlay) that ask them to identify themselves as a beacon for a given hash function and target location. This allows secret servlets to locate beacons for a given hash function and inform those beacons of their identity as secret servlet.

Under the Chord service, all nodes that are beacons under the same hash function can be made aware aware of each other's existence.[2] We assume that SOS will utilize a finite set of possible hash functions per target. We do not require that all beacons know about all secret servlets, but that each beacon knows about at least one active secret servlet. By spreading the set of usable hash functions across the secret servlets, having each secret servlet inform a beacon for each hash function it contains, and having all beacons under the same hash function share their secret servlet information, it is easy to show that all beacons have access to at least one secret servlet. Thus, our secure path from the confirmed source point to the target is complete.

3.5 Summary of Architecture

To summarize, the sequence of operations in the SOS architecture consists of the following steps:

1. A site (target) selects a number of SOS nodes to act as secret servlets; that is, nodes that are allowed to forward traffic to that site. Routers in the perimeter of the site are instructed to only allow traffic from these servlets to reach the internal of the site's network.

2. When an SOS node is informed that it will act as a secret servlet for a site (and after verifying the authenticity of the request), it will compute the key k for each of a number of well-known consistent hash functions, based on the target site's network address range. Each of these keys will identify a number of overlay nodes that will act as beacons for that target site.

3. Having identified the beacons, the servlets will contact them and notify them of their function. Beacons, after verifying the validity of the request, will store the necessary information to forward traffic for that site to the appropriate servlet. Beacons of the same key may share information about the set of available servlets for a target.

4. A source that wants to communicate with the target contacts an overlay access point (OAP). After authenticating and authorizing the request, the OAP routes all traffic from the source to the target to one of the beacons. The OAP (and all subsequent hops on the overlay) can route the packet to an appropriate beacon in a distributed fashion using Chord by using computation of the hash function(s) over the target's address to identify the next hop on the overlay.

5. The beacon then routes the packet to a secret servlet that then routes the packet (through the filtering) to the target.

This scheme is robust against DoS attacks because:

- If an access point is attacked, the confirmed source point can simply choose an alternate access point by which it enters the overlay.

- If a node within the overlay is attacked, the node simply exits the overlay and the Chord service self-heals, providing new paths to (potentially new sets of) beacons. Furthermore, no node is more important or sensitive than others — even beacons can be attacked and are allowed to fail.

[2]The set of beacons form an ordered list, where each beacon knows the existence of the next beacon on the list. A trivial extension could make this relation bi-directional if necessary for the purposes of exchanging secret servlet information.

- If a secret servlet is attacked (either due to a lucky random hit or somehow its identity was compromised), the secret servlet (which can still send traffic outward) can notify the target and the target can choose alternate secret servlets.
- If a secret servlet's identity is discovered and attacks arrive at the target with the source IP address of some secret servlet, the target can choose an alternate set of secret servlets.

4 Modeling DoS and Performance Analysis of SOS

In this section we develop simple analytical models that describe DoS attacks, and describe how the SOS architecture helps prevent the possibility of successful DoS attacks. DoS attacks are launched through compromised nodes across the network. Nodes can be compromised, for instance, by an email virus, such that the attack is triggered at a later point in time (*e.g.,* when a compromised node executes the attack code via an email client). While the spread of the virus can be modeled as an epidemiological process[KW91], the launch of the attack by these compromised nodes can be viewed as independent events across compromised clients. We examine an attack from two directions. First, we look at the *attack severity*: the effect that a certain level of attack traffic has on a node's ability to withstand attack as a function of the node's capacity and processing power. We show that, by protecting the edge via filtering, SOS forces attacks into the core of the network. We show that the amounts of traffic needed to bring down a node increase significantly as a result. Next, we look at the *attack success rate*: because SOS can quickly switch a path from a set of attacked nodes to a set of un-attacked nodes, an attacker must successfully attack a very large set of nodes in order to successfully shut down communication to the target. We show that unless the attacker can shut down a very large set of nodes, the likelihood of a successful attack is slim. From these two properties of SOS, it follows that bringing down an SOS-protected target is essentially impossible.

4.1 Attack Severity

We measure attack severity in a scenario in which several compromised zombie nodes, distributed over the network, launch attacks on a target node. We show that attacks that start at random times will overpower routers with low bandwidth capabilities much easier than those with high bandwidth capabilities.

As a simple first approximation, we could view the arrival of the attack traffic from such clients as a Poisson process, with an arrival rate λ_a attacks per unit time. Each attacking client is assumed to use up b_a units of resources (typically bandwidth) from a target while the attack is in progress. We also assume that the duration of attacks from such clients is exponentially distributed, with mean $1/\mu_a$ (the attacks could terminate for a number of reasons, for instance discovery and shutdown of compromised clients by users/local system administrators or discovery by some trace-back mechanism and shutdown by access network filtering). We also assume that legitimate traffic arrives at the node with rate λ_l, requiring b_l units of resource and a mean holding time $1/\mu_l$. Let us assume that the target node has C_t units of resource available. When all the resources get tied up, arriving requests, either legitimate or attack, are denied service. We then say that a DoS attack is successful.

The system model can now be abstracted into a Stochastic Knapsack [Ros95] framework. In a Stochastic Knapsack, C_t is the total amount of resources available at the server, and each arriving connection is mapped into an arriving call of class m with resource requirement b_m and mean holding time $1/\mu_m$. Calls in each class arrive at a rate λ_m. The knapsack always admits an arriving object when there is sufficient room. In our model, the probability of a successful DoS attack is the blocking probability corresponding to the class of legitimate traffic.

Let n_m denote the number of class-m objects in the knapsack. Then the total amount of resource utilized by the objects in the knapsack is $\mathbf{b} \cdot \mathbf{n}$, where $\mathbf{b} := (b_1, b_2, \ldots, b_M)$ and $\mathbf{n} := (n_1, n_2, \ldots, n_M)$. We define the process in terms of the state space of the different class-m objects, *i.e.,* let

$$\mathcal{K} := \{\mathbf{n} \in \mathcal{I}^m : \mathbf{b} \cdot \mathbf{n} \leq C_t\}$$

Formally, the knapsack admits an arriving class-m object if $b_m \leq C_t - \mathbf{b} \cdot \mathbf{n}$. Let $\mathcal{K}_m$ be the subset of such states, *i.e.,*

$$\mathcal{K}_m := \{\mathbf{n} \in \mathcal{K} : \mathbf{n} \leq C_t - b_m\}$$

The blocking probability B_m for a class-m call under Poisson arrival assumption is then given by [Ros95]

$$B_m = 1 - \frac{\sum_{\mathbf{n} \in \mathcal{K}_m} \prod_{j=1}^{M} \rho_j^{n_j}/n_j!}{\sum_{\mathbf{n} \in \mathcal{K}} \prod_{j=1}^{M} \rho_j^{n_j}/n_j!} \tag{1}$$

where $\rho_j = \lambda_j/\mu_j$.

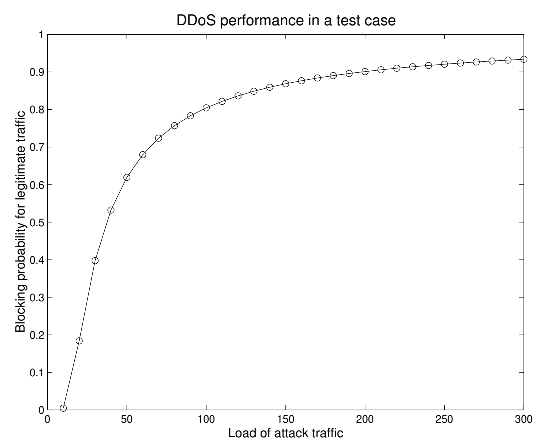


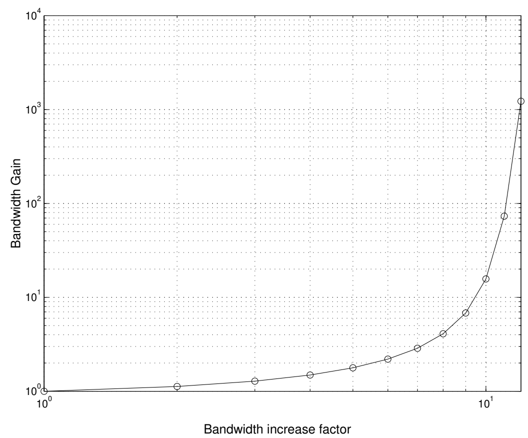
Figure 2: Blocking probability for legitimate traffic as a function of attack traffic load.

As an illustrative example, we consider a simple case where we have only two classes of customers, one corresponding to the DoS attacks and the other to legitimate traffic.[3] We assume that an individual call in each class uses up the same amount of bandwidth (motivated by the idea that the compromised clients come from the same population as the legitimate users). For a DoS attack to be successful, the load level (ρ_j) for the class of attack traffic has to be significantly higher than that of legitimate traffic. We construct a test scenario where the target node has 20 units of resource available, both the attack and legitimate traffic utilize one unit of resource and $\lambda/\mu(\rho)$ for the legitimate traffic is 1. In Figure 2, we plot the probability that a legitimate connection is denied service as a function of ρ of of attack traffic.

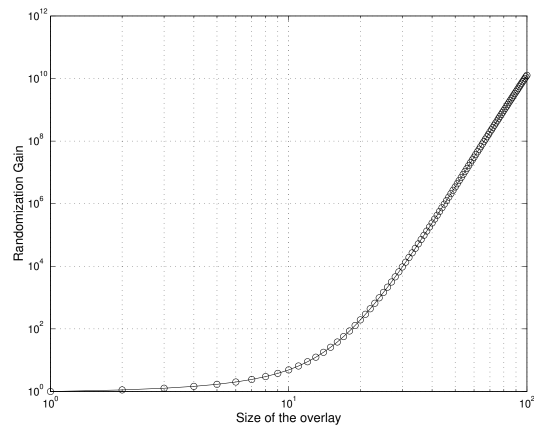
As we can observe, under our test scenario, where $\rho = 200$ for the attack traffic will cause 90% of the legitimate traffic to be denied service. Under a massive attack, if the attack load rises to 10000, 99.8% of legitimate traffic is denied service. Now we consider the effects of two key features of the SOS architecture.

[3] In a more accurate or generalized model, we can classify the various clients according to their bandwidth capabilities, more specifically their network access types like DSL, Cable, T1, Dialup, *etc.* This would not change the nature of the results we present.

First, when we push the attack point perimeter into the interior of the core, then the traffic handling capability of the attacked node increases (core routers can handle OC192 line speeds per interface, compared to 155Mbps capabilities of a typical high speed edge router). We consider the case where the attack traffic load in our test scenario is 200, and we re-compute the blocking probability for legitimate traffic as we increase the capacity of the node by a factor r, *i.e.,* $C_{t_{new}} = r \times C_{t_{old}}$. We denote the ratio of the old blocking probability with the new blocking probability as the Bandwidth Gain (BG) of the system. In Figure 3(a), we plot the BG of the system as a function of r. As can be observed, a bandwidth increase by a factor of 12 brings about a reduction in the blocking probability by three orders of magnitude.



(a) Increasing the capacity of the attacked node



(b) Increasing the anonymity of the attacked node

Figure 3: Performance gains with SOS.

Next, we study the effects of anonymizing the attacked node. Thus, if the attacker does not know the identity of the secret servlet for a particular target, then the attacks would be launched randomly onto the overlay. Only a fraction of those attacks would reach the target servlet. Thus, the effective arrival rate of the attacks would become $\lambda_a \times f$, where f is the fraction of the secret servlets in the SOS for a particular node. We again compute the ratio of the old probability with the new blocking probability and denote it as the Randomization Gain (RG) of the system. In Figure 3(b) we plot the RG of the system as a function of the number of nodes in the overlay (as the number of nodes in the overlay increase from left to right, a smaller and smaller fraction of the traffic reaches the target node). Placing the target node randomly in a group of 30 brings down the probability of attack by 4 orders of magnitude.

The preceding analysis demonstrates that bringing down targeted nodes in the SOS architecture becomes very difficult for a potential attacker. However, an attacker might decide to attack an arbitrary node (or nodes) on the overlay with the intention of a random DoS attack. Although the difficulty of bringing down even a single (high capacity) node on the overlay down is considerable, in the next section we show how the redundancy of SOS helps in preventing such random attacks.

4.2 Attack Success Rate

We now evaluate the likelihood that an attacker (or set of attackers) is able to prevent communication between a particular pair of (arbitrarily chosen) client and target. We investigate two fundamental scenarios:

- **Static Case:** Here, we assume that an attacker chooses a fixed set of nodes within the overlay upon which it will launch an attack. We allow the client's communication path to adapt to these attacks and find a route if one indeed exists. We present results where we consider the flow in isolation, where an attacker brings down a subset of nodes, and explore whether this prevents communication.
- **Dynamic Case:** Here, we assume that when an attacker attacks a node, the self-healing properties of SOS are able to repair the attack, forcing the attacker to then look elsewhere.

4.2.1 Static Case

Our analysis begins by considering the following problem: suppose some subset of nodes in the overlay are able to take on specific tasks for a given target, T. Let $\{S_1(T), S_2(T), \cdots, S_s(T)\}$ be the set of secret servlets with $U_s = |\{S_i(T)\}|$, $\{A_1(\mathcal{S}), \cdots, A_a(\mathcal{S})\}$ be the set of access points that can be used by source point $\mathcal{S}$ with $U_o = |\{A_i(\mathcal{S})\}|$, and $\{B_1(T), \cdots, B_b(T)\}$ be the set of beacons used to receive transmissions from $\mathcal{S}$ headed toward T: $U_b = |\{B_i(T)\}|$ is a function of the number of hash functions issued by T as well as the amount of redundancy implemented within Chord for each hash function.

For our initial analysis, we assume that a session between $\mathcal{S}$ and T can proceed so long as there exists an available access point, an available beacon, and an available secret servlet. This assumes that all beacons are aware of all secret servlets. The analysis is easily extended for the case where this assumption does not hold. We also assume that the selection of nodes to perform various duties is done independently, such that a node could simultaneously act as any combination of access point, beacon, and secret servlet. We assume that all nodes implement (and hence can be part of the path) of the Chord routing service. Thus, Chord will be able to route effectively even if only one node remains in the overlay, though the node would have to simultaneously be the access point, beacon, and secret servlet.

Let $P_h(a, b, c)$ be the probability that a set of b nodes selected at random from $a \geq b$ nodes contains a specific subset of $c \leq b$ nodes. It is easy to show that $P_h(a, b, c) = \binom{b}{c} / \binom{a}{c}$.[4]

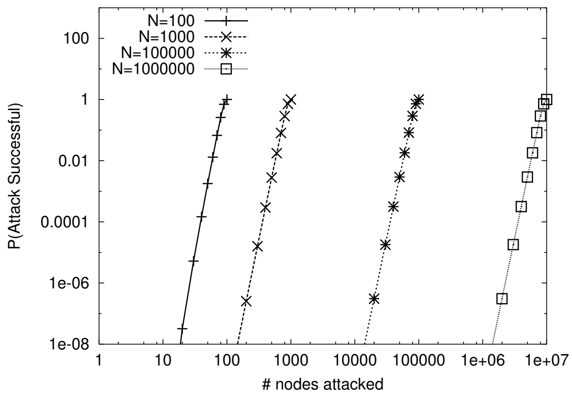
Let n_a be the number of nodes that the attacker attacks. Let $U_{\mathcal{S},T}$ be a random variable that equals 1 if $\mathcal{S}$ can reach T during an ongoing attack and 0 otherwise.

$$\begin{aligned}\Pr[U_{T,\mathcal{S}} = 1] = {} & (1 - P_h(N, n_a, U_s)) \cdot \\ & (1 - P_h(N, n_a, U_b)) \cdot (1 - P_h(N, n_a, U_o))\end{aligned}$$

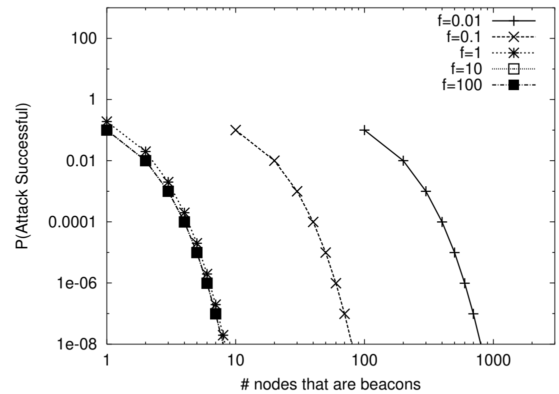
Figure 4 plots the likelihood of an attack succeeding at shutting down access to a site in the static case. In Figure 4(a) we hold U_s, U_b, and U_o all fixed at 10 and vary n_a along the x-axis. These numbers are quite conservative: we restrict the source's entry to only 10 possible access points and allow at most 10 beacons and secret servlets to service its needs. An increase in any of these numbers decreases the probability of a successful attack. The y-axis plots the probability of a successful attack, with the different curves representing different values of N, the total number of nodes in the overlay system. In Figure 4(b), we hold N fixed at 10,000 and n_a fixed at 1000. We vary U_b along the x-axis, and again plot the probability of a successful attack on the y-axis. The different curves represent the probabilities for different values of U_s, where $f = U_s / U_b$.

From these figures, we see that the likelihood of an attack successfully terminating communication between $\mathcal{S}$ and T is negligible unless the attacker can simultaneously bring down a significant fraction of nodes in the network. For instance, Figure 4(a) demonstrates that when only ten nodes act as beacons, ten nodes act as secret servlets, and ten nodes act as access points, for an attack to be successful in one out of ten thousand attempts, approximately forty percent of the nodes in the overlay must be attacked simultaneously.

[4] This follows from an algebraic reduction of $P_h(a, b, c) = \binom{a-c}{b-c} / \binom{a}{b}$.



(a) Varying number of attackers and nodes in the overlay



(b) Varying number of beacons and secret servlets

Figure 4: Attack success probability for the Static case.

Similarly, Figure 4(b) show that the likelihood of a successful attack is significant only when either the number of secret servlets or the number of beacons is small, but the numbers needed to force attacks to be successful beneath miniscule probabilities are quite small. In summary, long-term static attacks upon a moderately-provisioned SOS are unlikely.

4.2.2 Dynamic Case

In the Dynamic case, let the attacker have enough bandwidth resources to bring down K nodes. The attacks proceed in the following manner: The attacker attacks K nodes, Chord self-heals after an exponentially distributed time $1/\mu_{heal}$, the attacker realizes that and launches an attack on another node. The launch of the subsequent attacks is modeled as a Poisson process with rate λ_{attack}. We can model this system as a classical $M/M/\infty//K$ queueing system, *i.e.,* one with a finite number of (K) customers and an infinite number of servers. The average number of customers actively receiving service at any time,$\bar{N}$, is given by [Kle75]

$$\bar{N} = \frac{K\lambda_{attack}/\mu_{heal}}{1 + \lambda_{attack}/\mu_{heal}} \tag{2}$$

$\bar{N}$ is strictly less than K. In Figure 5 we plot the effect of the self-healing process. The effective K ($\bar{N}$) is plotted as a function of μ_{heal}. As we can observe, the faster the self-healing process as compared to attack arrivals, the lower $\bar{N}$. $\bar{N}$ represents the average number of nodes under attack at any given time. Thus, the effectiveness of the attacker is reduced considerably by the self-healing properties of Chord as fewer nodes come under simultaneous attack. The faster Chord self-heals, the more ineffective the attack is.

5 Implementation of SOS

One particularly attractive feature of the SOS architecture is that it can be implemented in a very straightforward manner, using existing software. In particular, the following components are currently available and in use:

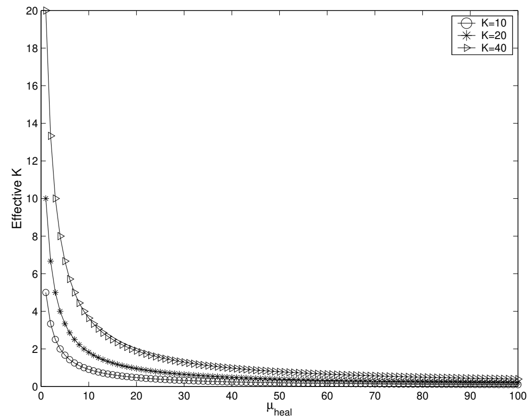


Figure 5: Benefits of the self-healing property of Chord.

- *Filtering:* all high and medium-range (both in terms of performance and price) routers, as well as most desktop and server operating systems, offer some high-speed packet classification scheme that can be used to implement the target perimeter filtering. A simplified version of [IB02] can be used by the target to inform its perimeter routers of changes in the set of allowed secret servlets.

- *Authentication and authorization of sources:* practically all commercial and free operating systems include an implementation of IPsec[KA98]. IPsec is a set of protocols that can be used to establish cryptographic keys and other relevant parameters between a pair of hosts, and then protect (encrypt and integrity-protect) the traffic between them. As described in [BIK01], the conditions under which access to the overlay is allowed can be efficiently encoded in public key certificates. Thus, it is possible to provision and manage access control for a large SOS infrastructure with minimal overhead in terms of performance, storage, and synchronization requirements.

  More specifically, each authorized source is given a certificate by a target authorizing that source to use the SOS infrastructure to send traffic to the target. In the process of authenticating to an access point (via the IPsec authentication protocol, currently IKE[HC98]), the source provides this certificate to the access point. The access point can both authenticate the source (by verifying a cryptographic signature) and confirm that the source is allowed to send traffic to the target. Notice that access points need not store any access control policies. The certificates are used to "remind" access points of the relevant access control policies; once a communication is torn down, the access point can "forget" the relevant policy.

- *Tunneling:* once traffic has entered the overlay network, it needs to be forwarded to other SOS nodes towards the beacon, and from there to the secret servlets. Standard traffic tunneling techniques[Ioa93, ABKM01] and protocols can be used to this end: IP-in-IP encapsulation[Per96], GRE encapsulation[FLH+00], or IPsec in "tunnel mode". Furthermore, traffic inside the overlay network can take advantage of traffic prioritization schemes such as MPLS or DiffServ, if they are made available by the infrastructure providers. The routing decisions inside the overlay network are based on a Chord-like mechanism[SMK+01].

We envision the overlay nodes to be a mix of routers and high-speed end systems. In particular, since IP tunneling is a lightweight operation, it is conceivable that SOS functionality would be offered by service providers without adversely affecting the performance of their networks. The access points to the overlay network can be a mix of routers and high-speed end systems (with appropriate cryptographic acceleration hardware to boost performance). The access points and secret servlets could also act as "charging" points, if SOS-like functionality was offered on a commercial basis. Finally, since overlay nodes are only called upon to do encapsulated packet forwarding, cross-provider collaboration[5] is a fairly straightforward proposition, compared to controlled exposure of the filtering mechanism between different providers.

6 Discussion

Our study of SOS is admittedly in its early stages. There are several issues that need addressing for the service to have a viable impact within the Internet. In this section, we discuss current limitations and suggest directions for future research.

Attacks from inside the overlay: We have assumed that no malicious users can successfully bypass our protection perimeter. However, in practice, security management oversights or development bugs could lead to situations where breaches occur. An evaluation of the potential damages that can be done from the inside and approaches to limit these damages warrants further investigation.

A shared overlay: We have presented SOS as a means to permit communication from a single confirmed source point to a single target. The architecture should easily scale to handle numerous confirmed source points transmitting to multiple targets. We note that in its current form, state for each target must be maintained at the secret servlets and beacons that support those targets as well as at access points (to confirm a source point's right to contact the target). Scalability is improved by limiting the set of access points, secret servlets and beacons that offer support to a given target. However, this makes the service more prone to DoS attacks. The overlay becomes more efficient at protecting users from DoS attacks as it grows. Hence, it would be of interest for multiple organizations to utilize a shared overlay. However, this could increase the likelihood of the overlay being compromised from the inside. We intend to investigate some form of sandboxing that could be constructed within the shared overlay such that a breach in one organization's security system would not lead to breaches in other security systems.

Timely delivery: To achieve security, SOS forces traffic through a series of overlay points that perform different tasks. We suspect that the latency across the path is far from minimal. It would be of interest to see if there are any "shortcuts" through the overlay that do not compromise security, or to extend the architecture such that it contains a "knob" that allows users to trade levels of security with timely delivery.

Analysis: Our analysis presented here is preliminary. More sophisticated means of analyzing SOS, either via a more detailed mathematical model or through prototype and experimentation are needed to better understand its operation.

7 Conclusion

In this paper, we addressed the problem of securing a communication service on top of the existing IP infrastructure from DoS attacks. It is envisioned that such a service would be offered, among others, to emergency teams in the aftermath of a disaster, to facilitate communication between the teams and various agencies and organizations over the Internet.

We attack the problem with a *proactive* mechanism, which is composed of aggressive packet filtering in a site's network periphery, an *overlay network* that can self-heal during (and after) a DoS attack, and a scalable access control mechanism that allows legitimate users to use the overlay network. We call this architecture *Secure Overlay Services*, or *SOS*.

[5]While it is not strictly necessary that different service providers connect their overlay networks, doing so would allow them to exploit the benefits of scale described in Section 4.

Through simple analytical models we show that DoS attacks directed against any part of the SOS infrastructure have negligible probability of disrupting the communication between two parties: for instance, when only ten nodes act as beacons, ten nodes act as secret servlets, and ten nodes act as access points, for an attack to be successful in one out of ten thousand attempts, approximately forty percent of the nodes in the overlay must be attacked simultaneously. Furthermore, the resistance of a SOS network against DoS attacks increases greatly with the number of nodes that participate in the overlay. Implementing an SOS infrastructure is fairly straightforward and can be done using almost exclusively off-the-shelf protocols and software.

We believe that our approach is a novel and powerful way of countering DoS attacks, especially in service-critical environments. While there remain several issues to be solved, our work should encourage researchers to investigate proactive approaches in addressing the DoS problem.

References

[ABKM01] D.G. Andersen, H. Balakrishnan, M.F. Kaashoek, and R. Morris. Resilient Overlay Networks. In *Proc. of the 18th Symposium on Operating Systems Principles (SOSP)*, October 2001.

[BIK01] M. Blaze, J. Ioannidis, and A.D. Keromytis. Trust Managent for IPsec. In *Proc. of Network and Distributed System Security Symposium (NDSS)*, pages 139–151, February 2001.

[Cla88] D. D. Clark. The Design Philosophy of the DARPA Internet Protocols. In *Proc. of SIGCOMM 1988*, pages 106–114, 1988.

[DFS01] D. Dean, M. Franklin, and A. Stubblefield. An Algebraic Approach to IP Traceback. In *Proc. of the Network and Dsitributed System Security Symposium (NDSS)*, pages 3–12, February 2001.

[DKM+01] F. Dabek, F. Kaashoek, R. Morris, D. Karger, and I. Stoica. Wide-area cooperative storage with cfs. In *Proceedings of ACM SOSP'01*, Banff, Canada, October 2001.

[FLH+00] D. Farinacci, T. Li, S. Hanks, D. Meyer, and P. Traina. Generic routing encapsulation (GRE). Request for Comments 2784, Internet Engineering Task Force, March 2000.

[HB96] L.T. Heberlein and M. Bishop. Attack Class: Address Spoofing. In *Proceedings of the 19th National Information Systems Security Conference*, pages 371–377, October 1996.

[HC98] D. Harkins and D. Carrel. The Internet Key Exchange (IKE). Request for Comments (Proposed Standard) 2409, Internet Engineering Task Force, November 1998.

[IB02] J. Ioannidis and S. M. Bellovin. Implementing Pushback: Router-Based Defense Against DDoS Attacks. In *Proc. of the Network and Distributed System Security Symposium (NDSS)*, February 2002.

[Ioa93] J. Ioannidis. *Protocols for Mobile Networking*. PhD thesis, Columbia University, New York, 1993.

[KA98] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. Request for Comments (Proposed Standard) 2401, Internet Engineering Task Force, November 1998.

[Kle75] Leonard Kleinrock. *Queueing Systems, Volume I: Theory*. Wiley-Interscience, 1975.

[KW91] J. O. Kephart and S. R. White. Directed-Graph Epidemiological Models of Computer Viruses. In *Proceedings of Computer Society Symposium on Research in Security and Privacy*, pages 343–359, 1991.

[MVS01] D. Moore, G. Voelker, and S. Savage. Inferring Internet Denial-of-Service Activity. In *Proc. of the 10th USENIX Security Symposium*, pages 9–22, August 2001.

[Per96] C. Perkins. IP encapsulation within IP. Request for Comments 2003, Internet Engineering Task Force, October 1996.

[Ros95] Keith W. Ross. *Multiservice Loss Models for Broadband Telecommunication Networks*. Springer-Verlag, 1995.

[SCWA99] S. Savage, N. Cardwell, D. Wetherall, and T. Anderson. TCP Congestion Control with a Misbehaving Receiver. *ACM Computer Communications Review*, 29(5):71–78, October 1999.

[SKK+97] C. Schuba, I. Krsul, M. Kuhn, E. Spafford, A. Sundaram, and D. Zamboni. Analysis of a denial of service attack on tcp. In *IEEE Security and Privacy Conference*, pages 208–223, May 1997.

[SMK+01] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of ACM SIGCOMM'01*, San Diego, CA, August 2001.

[SRC84] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end arguments in System Design. *ACM Transactions on Computer Systems*, 2(4):277–288, November 1984.

[SWKA00] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Practical Network Support for IP Traceback. In *Proc. of the 2000 ACM SIGCOMM Conference*, pages 295–306, August 2000.

[SWKA01] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Network Support for IP Traceback. *ACM/IEEE Transactions on Networking*, 9(3):226–237, June 2001.

[Tea96] Software Engineering Institute Computer Emergency Response Team. CERT Advisory CA-96.21: TCP SYN Flooding and IP Spoofing Attacks. Technical report, SEI, Sept 1996. ftp://info.cert.org/pub/cert_advisories/CA-96.21.tcp_syn_flooding.