SOS: An Architecture For Mitigating DDoS Attacks

Angelos D. Keromytis, *Member, IEEE,* Vishal Misra, *Member, IEEE,* Dan Rubenstein, *Member, IEEE*

***Abstract*— We propose an architecture called *Secure Overlay Services (SOS)* that *proactively* prevents DoS attacks, geared toward supporting Emergency Services or similar types of communication. The architecture uses 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 forwarding 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. Our performance measurements using a prototype implementation indicate an increase in end-to-end latency by a factor of 2 for the general case, and an average heal time of less than 10 seconds.

***Index Terms*— Denial of service attacks, overlay networks, peer-to-peer networks, access control, packet filtering.**

I. INTRODUCTION

A SECURE system meets or exceeds an application-specified set of security policy *requirements*. For example, in message delivery, the high-level requirements may be that the correct information gets to the right person, in the right place, *at the right time*. The details of "right" are determined by the application's needs. For example, during a crisis, the network can be used to carry communications between widely dispersed "static" sites (*e.g.,* various federal, state, and city agencies) and (semi-) roaming stations and users. Similarly, timely message delivery is crucial for battlefield or stock-trading tasks. Traditional security mechanisms have addressed the first two parts of this informal definition of security, but largely ignored the timeliness or service guarantee issue. One threat to timely data delivery in a public network such as the Internet is denial of service (DoS) attacks: these attacks overwhelm the processing or link capacity of the target site (or routers that are topologically close) by saturating it (them) with bogus packets. Such attacks can seriously disrupt legitimate communications at minimal cost and danger to the attacker, as has been demonstrated repeatedly in recent years.

In the SOS architecture [1] we address the problem of securing communication in today's existing IP infrastructure from denial of service (DoS) attacks, where the communication is between a pre-determined location and a set of well-known users, located anywhere in the wide-area network, who have authorization to communicate with that location. We focus our efforts on protecting a site that stores information that is difficult to replicate due to security concerns or due to its dynamic nature. An example is a database that maintains timely or confidential information such as building structure reports, intelligence, assignment updates, or strategic information. We assume that there is a pre-determined set of clients scattered throughout the network who require (and should have) access to this information, from anywhere in the network.

Contrary to the other approaches we review in Section VI, which are reactive, our approach is *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 (potentially as small as 2-3 addresses) that are "approved" at any particular time is kept secret so that attackers cannot use them to pass through the filter. These addresses are picked from among those within a distributed set of nodes throughout the wide area network, that 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 from attackers) locations, which can then forward the validated traffic through the filtering routers to the target. The two main principles behind our design are: (i) elimination of communication pinch-points, which constitute attractive DoS targets, via a combination of filtering and overlay routing to obscure the identities of the sites whose traffic is permitted to pass through the filter, and (ii) the ability to recover from random or induced failures within the forwarding infrastructure or within the secure overlay nodes.

We discuss how to design the overlay such that it is secure with high probability, given that attackers have a large but finite set of resources to perform the attacks. The attackers can also know the IP addresses of the nodes that participate in the overlay and of the target that is to be protected, as well as the details of the operation of protocols used to perform the forwarding. However, we assume that (a) the attacker does not have unobstructed access to the network core, and (b) the attacker cannot severely disrupt large parts of the backbone.

Our architecture leverages heavily off of previous work on IP security [2], IP router filtering capabilities, and novel

Manuscript received November 15, 2002; revised June 1, 2003. This work is supported in part by DARPA contract No. F30602-02-2-0125 (FTN program) and by the National Science Foundation under grant No. ANI-0117738 and CAREER Award No. ANI-0133829, with additional support from Cisco Corporation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Angelos D. Keromytis is with the Computer Science Department, Columbia University. Email: angelos@cs.columbia.edu

Vishal Misra is with the Computer Science Department, Columbia University. Email: misra@cs.columbia.edu

Dan Rubenstein is with the Electrical Engineering Department, Columbia University. Email: danr@ee.columbia.edu

approaches to routing in overlays [3] and peer-to-peer (P2P) networks [4], [5]. To the extent possible, we strive to use existing systems and protocols, rather than invent our own. Our resulting system is in some ways similar to the Onion Routing architecture [6] used for anonymous communications.

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 obtained by an attacker through the exploitation of 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 that accesses the overlay through a small subset of overlay nodes. We use our prototype implementation with PlanetLab, a distributed infrastructure for experimentation on overlay networks, to measure the increase in end-to-end latency. We determine that using SOS increases the latency by a factor of 2, which we consider acceptable in comparison to the latency or lack of communication when the when a debilitating DDoS attack is successfully launched. Furthermore, we experimentally determine that the overlay can heal itself within 10 seconds of being targeted by such an attack.

II. Architecture Description

The goal of the SOS architecture is to allow communication between a *confirmed user* and a *target*. By confirmed, we mean that the target has given prior permission to this user. Typically, this means that the user's packets must be authenticated and authorized by the SOS infrastructure before traffic is allowed to flow between the user through the overlay to the target. We use the techniques we developed in [2] for this purpose. While we focus on the communication to a single target, the architecture is easily extended to simultaneously protect unicast communications destined to different targets. Both peers can use the SOS infrastructure to protect bidirectional communications; this is particularly important for "static" sites (*e.g.,* two branches of the same company). For mobile clients the reverse direction's traffic (from the target site to the client) can be sent directly over the Internet, or it can also use the SOS infrastructure.

SOS is a network overlay, composed of nodes that communicate with one another atop the underlying network substrate. Often, nodes will perform routing functionality to deliver messages (packets) from one node in the overlay to another. We assume that the set of nodes that participate in the overlay is known to the public and hence also to any attacker. In effect, no node's identity is kept hidden. However, certain roles that an overlay node may assume in the process of delivering traffic are kept secret from the public. Keeping participation information of certain nodes hidden from the public could be a means of providing additional security, but is not required.

Attackers in the network 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, to effectively shut down that node's ability to receive 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 saturate the target site. We assume attackers are smart enough to exploit features of the architecture that are made publicly available, such as the set of nodes that form the overlay. In this paper, we do not specifically consider how to protect the architecture against attackers who can infiltrate the security mechanism that distinguishes legitimate traffic from (illegitimate) attack traffic: we assume that communications between overlay nodes remain secure so that an attacker cannot send illegitimate communications, masking them as legitimate. In addition, it is conceivable that more intelligent attackers could monitor communications between nodes in the overlay and, based on observed traffic statistics, determine additional information about the current configuration. Protecting SOS from such attackers is beyond the scope of this paper [7].

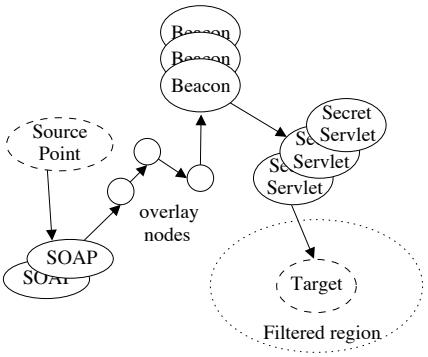


Fig. 1. Basic SOS architecture.

Figure 1 gives a high-level overview of the SOS architecture that protects a target node or site so that it only receives legitimate transmissions. In the discussion that follows, we first give a brief overview of the design process, and then develop the architecture piece by piece. The reader can refer back to the figure during the discussion.

A. Design Rationale

Fundamentally, the goal of the SOS infrastructure is to distinguish between authorized and unauthorized (or, more generally, unverified) traffic. The former is allowed to reach the destination, while the latter is dropped or is rate-limited. Thus, at a very basic level, we need the functionality of a firewall "deep" enough within the network so that the access link to the target is not congested. This imaginary firewall would perform access control by using protocols such as IPsec.

However, traditional firewalls themselves are susceptible to DoS attacks. One way to address this problem is to replicate the firewall functionality, in a manner similar to that described in [8]. To avoid the effects of a DoS attack against the firewall connectivity, we need to distribute these instances of the firewall across the network. In effect, we are "farming out" the expensive processing (such as cryptographic protocol handling) to a large number of nodes. However, firewalls depend on topological restrictions in the network to enforce access control policy. Since our distributed firewall has performed the access control step, it would seem obvious that all we need around the target is a router that is configured to only let through traffic forwarded to it by one of the firewalls.

However, a security system cannot depend on the identity of these firewalls to remain secret. Thus, an attacker can launch a DoS attack with spoofed traffic purporting to originate from one of these firewalls. Notice that, given a sufficiently large group of such firewalls, we can select a very small number of these as the designated authorized forwarding stations: only traffic forwarded from these will be allowed through the filtering router, and we change this set periodically.

B. Architecture Overview

The forwarding of a packet within the SOS architecture, depicted in Figure 1, proceeds through five stages:

- A source point that is the origin of the traffic forwards a packet to a special overlay node called a SOAP that receives and verifies that the source point has a legitimate communication for the target.
- The SOAP routes the packet to a special node in the SOS architecture that is easily reached, called the beacon.
- The beacon forwards the packet to a "secret" node, called the secret servlet, whose identity is known to only a small subset of participants in the SOS architecture.
- The secret servlet forwards the packet to the target.
- The filter around the target stops all traffic from reaching the target except for traffic that is forwarded from a point whose IP address is the secret servlet.

In the following discussion, we motivate why the SOS architecture requires the series of steps described above.

C. Protecting the Target: Filtering

In the current Internet, knowledge of the target's network identifier (IP address) allows an attacker to bombard the target location with packets that originate from compromised locations throughout the Internet. To prevent these attacks, a *filter* can be constructed that drops illegitimate packets at some point in the network, such that the illegitimate traffic does not overwhelm routing and processing resources at or near the target. We assume that the filter can be constructed so that attackers do not have access to routers inside the filtered region (*i.e.,* they cannot observe which source addresses can proceed through the filter). Past history indicates that it is significantly 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).

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 can still traverse the others and ultimately reach the target. Essentially, we assume that the filter can be constructed locally around the target to prevent a bombardment of illegitimate traffic, while at the same time allowing legitimate, filtered traffic to successfully reach the target. Such filters need to be established at the ISP's Point of Presence (POP) routers that attach to the ISP backbone.

D. Reaching Well-filtered Target

Under the filtering mechanism described previously, legitimate users can reach the target by setting the filter around the target to permit only those IP addresses that contain legitimate users. This straightforward approach has two major shortcomings. First, whenever a legitimate user moves, changes IP address, or ceases to be legitimate, the filter surrounding the target must be modified. Second, the filter does not protect the target from traffic sent by an illegitimate user that resides at the same address as a legitimate user, or (more importantly) from an illegitimate user that has knowledge about the location of a legitimate user and spoofs the source address of its own transmissions to be that of the legitimate user.

A first step in our solution is to have the target select a subset of nodes, N_s, that participate in the SOS overlay to act as *forwarding proxies*. The filter only allows packets whose source address matches the address of some overlay node $n \in N_s$. Since n is a willing overlay participant, it is allowed to perform more complex verification procedures than simple address filtering and use more sophisticated (and expensive) techniques to verify whether or not a packet sent to it originated from a legitimate user of a particular target.

The filtering function that is applied to a packet or flow can have various levels of complexity. It is, however, sufficient to filter on the source address: the router only needs to let through packets from one of the few forwarding proxies. All other traffic can be dropped, or rate-limited. Because of the small number of such filter rules and their simple nature (source IP address filtering), router performance will not be impaired [9], even if we do not utilize specialized hardware.

This architecture prevents attackers with knowledge of legitimate users' IP addresses from attacking the target. However, an attacker with knowledge of the IP address of the proxy can still launch two forms of attacks: an attacker can breach the filter and attack the target by spoofing the source address of the proxy, or attack the proxy itself. This would prevent legitimate traffic from even reaching the proxy, cutting off communication through the overlay to the target.

Our solution to this form of attack is to *hide the identities of the proxies*. If attackers do not know the identity of a proxy, they cannot mount either form of attack mentioned above unless they successfully guess a proxy's identity. We refer to these "hidden" proxies as *secret servlets*.

E. Reaching a Secret Servlet

To activate a secret servlet, the target sends a message to the overlay node that it chooses to be a secret servlet, informing that node of its task. Hence, if a packet reaches a secret servlet and is subsequently verified as coming from a legitimate user, the secret servlet can then forward the packet through the filter to the target. The challenge at this point is constructing a routing mechanism that will route to a secret servlet while utilizing a minimal amount of information about its identity.

Here we take advantage of the dynamic nature and the high level of connectivity that exists when routing atop a network overlay. The connectivity graph of a network overlay consists of nodes which are the devices (*e.g.,* end-systems) that participate in the overlay, and edges which represent IP paths that connect pairs of nodes in the overlay. Unlike the underlying network substrate whose physical requirements limit the pairs of nodes that can directly connect to one another, network overlays have no such limits, such that an overlay edge is permissible between any pair of overlay nodes. This added flexibility and increased number of possible routes can be used to complicate the job of an attacker by making it more difficult to determine the path taken within the overlay to a secret servlet. In addition, since a path exists between every pair of nodes, it is easy to recover from a breach in communication that is the result of an attack that shuts down a subset of overlay nodes. The recovery involves having the overlay route around these nodes. The underlying assumption is that network core links cannot easily be shut down.

There exists a straightforward but costly solution to reaching a secret servlet without revealing the servlet's ID to the nodes that wish to reach it: have each overlay node that receives a packet randomly choose the next hop on the overlay to which it forwards a packet [10]. Eventually, the packet will arrive at a secret servlet that can then deliver it to the target.

F. Connecting to the Overlay

Legitimate users need not reside at nodes that participate in SOS. Hence, SOS must support a mechanism that allows legitimate traffic to access the overlay. For this purpose, we define a *secure overlay access point (SOAP)*. A SOAP is a node that will receive packets that have not yet been verified as legitimate, and perform this verification. This verification can be performed using off-the-shelf authentication protocols such as IPsec or TLS. Allowing a large number of overlay nodes to act as SOAPs increases the bandwidth resources that an attacker must obtain to prevent legitimate traffic from accessing the overlay. Effectively, SOS becomes a large distributed firewall [8] that discriminates between "good" (authorized) traffic from "bad" (unauthorized) traffic. By using a large number of topologically-distributed firewall instances, we increase the amount of resources (bandwidth) an attacker has to spend to deny connectivity to legitimate clients. Note that if an attacker manages to acquire a legitimate user's authorization material, he can use multiple SOAPs to mount a DDoS attack from inside the overlay. In that case, the secret servlet or the beacon can use a pushback-like mechanism [9] to ask the SOAPs to revoke the user's authorization.

Having a large number of SOAPs increases the robustness of the architecture to attacks, but complicates the job of distributing the security information that is used to determine the legitimacy of a transmission toward the target. One can imagine several ways in which SOAPs can be chosen. For instance, different users (IP address origins) can be mapped to different subsets of SOAPs. Given the relatively small number of nodes that SOS requires, as we shall see in Section III, a list of all SOS nodes may be publicized and used by all clients. We plan to investigate SOAP selection in future work.

G. Routing through the Overlay

Having each overlay participant select the next node at random is sufficient to eventually reach a secret servlet [10]. However, it is rather inefficient, with the expected number of intermediate overlay nodes contacted being $O(N/N_s)$ where N is the number of nodes in the overlay and N_s is the number of secret servlets for a particular target. Here, we discuss an alternative routing strategy in which, with only one additional node knowing the identity of the secret servlet, the route from a SOAP to the secret servlet has an expected path length that is $O(\log N)$. We use Chord [4], which can be viewed as a routing service that can be implemented atop the existing IP network fabric, *i.e.,* as a network overlay. Consistent hashing [11] is used to map an arbitrary identifier to a unique destination node that is an active member of the overlay.

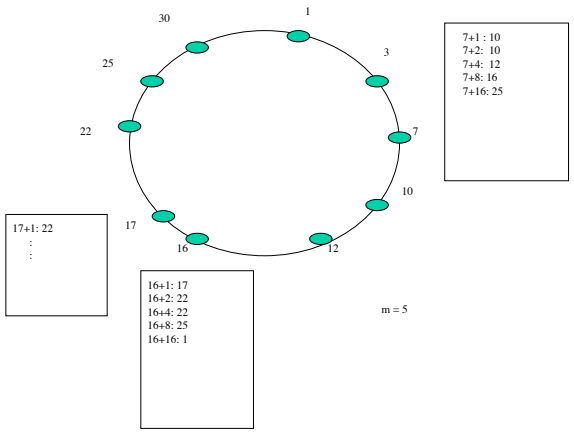


Fig. 2. Chord-based overlay routing.

In Chord, each node is assigned a numerical identifier (ID) via a hash function in the range $[0, 2^m]$ for some pre-determined value of m. The nodes in the overlay are ordered by these identifiers. The ordering is cyclic (*i.e.,* wraps around) and can be viewed conceptually as a circle, where the next node in the ordering is the next node along the circle in the clockwise direction. Each overlay node maintains a table that stores the identities of m other overlay nodes. The i^{th} entry in the table is the node whose identifier x equals or, in relation to all other nodes in the overlay, most immediately follows $x + 2^{i-1} \pmod{2^m}$, as shown in Figure 2. When overlay node x receives a packet destined for ID y, it forwards the

packet to the overlay node in its table whose ID precedes y by the smallest amount. In the example, if node 7 receives a packet whose destination is the identifier 20, the packet will route from 7 to 16 to 17, When the packet reaches node 17, the next node in the overlay is 22, and hence node 17 knows that 22 is responsible for identifier 20. Chord routes packets around the overlay "circle", progressively getting closer to the desired node, visiting $O(m)$ nodes. Typically, the hash functions used to map nodes to identifiers do not attempt to map two geographically close nodes to nearby identifiers. Hence, often two nodes with consecutive identifiers are geographically distant from one another within the network.

The Chord service is robust to changes in overlay membership, and each node's list is adjusted to account for nodes leaving and joining the overlay such that the above stated properties continue to hold. [12] discusses various security considerations for peer-to-peer networks that use distributed hash tables. Most of these do not apply here, since membership in the SOS overlay is "closed" — the clients and the targets are not considered part of the overlay, and can only interact with it through a well-defined interface that requires strong authentication and authorization.

SOS uses the IP address of the target as the identifier to which the hash function is applied. Thus, Chord can direct traffic from any node in the overlay to the node that the identifier is mapped to, by applying the hash function to the target's IP address. This node, to which Chord delivers the packet, is not the target, nor is it necessarily the secret servlet. It is simply a unique node that will be eventually be reached, regardless of the entry point. This node is called the *beacon*, since it is to this node that packets destined for the target are first guided. Thus, Chord provides a robust and reliable, while relatively unpredictable for an adversary, means of routing packets from an overlay access point to one of several beacons.

Finally, the secret servlet uses Chord to periodically inform the beacon of the secret servlet's identity. Should the servlet for a target change, the beacon will find out as soon as the new servlet sends an advertisement. If the old beacon for a target drops out of the overlay, Chord will route the advertisements to a node closest to the hash of the target's identifier. Such a node will know that it is the new beacon because Chord will not be able to further forward the advertisement. By providing only the beacon with the identity of the secret servlet, traffic can be delivered from any firewall to the target by traveling across the overlay to the beacon, then from the beacon to the secret servlet, and finally from the secret servlet, through the filtering router, to the target. This allows the overlay to scale for arbitrarily large numbers of overlay nodes and target sites. Unfortunately, this also increases the communication latency, since traffic to the target must be redirected several times across the Internet. If the overlay only serves a small number of target sites, traditional routing protocols or RON-like routing [3] may be sufficient. Other overlay routing mechanisms can also be used, *e.g.,* CAN [13].

H. Summary of Architecture

Before continuing on, we review the operational structure of SOS. A site (target) installs a filter in its immediate vicinity and then selects a number of SOS nodes to act as *secret servlets;* that is, nodes that are allowed to forward traffic through the filter to that site. Routers at the perimeter of the site are instructed to only allow traffic from these servlets to reach the internal of the site's network. These routers are powerful enough to filter on incoming traffic using a small number of rules without adversely affecting their performance.

When an SOS node is asked to 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. Each of these keys will identify a number of overlay nodes that will act as *beacons* for that target.

Having identified the beacons, the servlets or the target will contact and notify the beacons of the servlets' identities. Beacons verify the validity of the received information and store that information which is necessary to forward traffic for that target to the servlet.

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

Finally, the beacon routes the packet to a secret servlet that then routes it (through the filtering router) 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 to enter 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 over the re-formed overlay 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. Finally, if a secret servlet's identity is discovered and the servlet is targeted as an attack point, or attacks arrive at the target with the source IP address of some secret servlet, the target can choose an alternate set of secret servlets.

I. Redundancy

Having a single SOAP, beacon, or secret servlet weakens the SOS architecture, in that a successful attack on any one of these nodes can prevent legitimate traffic from reaching the target. Fortunately, each component is easily replicated within the architecture. Furthermore, an attack upon any of these components, once realized, is easily repaired.

Specifically, SOAP functionality is easily replicated. Any overlay node can act as a SOAP as long as it has the ability to check the legitimacy of a packet transmissions. If a SOAP is attacked, it can exit the overlay. A legitimate user attempting access need only contact another SOAP.

Furthermore, the target can choose multiple nodes as secret servlets and set the filter to allow packets from only these nodes to pass through the filter. If a secret servlet is attacked, or its identity breached such that attack traffic with a secret

servlet's source IP address can proceed through the filter, the target can remove the servlet whose identity is compromised from its set of servlets and modify its filter appropriately. A secret servlet under attack can also remove itself from the overlay until the attack terminates.

Finally, multiple nodes can act as beacons for a target by applying several hash functions (or several iterations of the same hash function) over the target identifier. In addition, if a beacon node is attacked, the node can remove itself from the overlay, and the Chord routing mechanism will heal itself such that a new node will act as a beacon for that hash function. If the former beacon cannot communicate the secret servlet information to the new beacon, then the new beacon must wait for the secret servlet to contact it again (as part of a keep-alive protocol) with its identity.

We note that when there are multiple beacons and secret servlets, every beacon should know the identity of at least one secret servlet so that the packets that each beacon receives can be forwarded onward to a secret servlet. Thus, each hash function is used by at least one secret servlet.

A last word on redundancy: since the secret servlets use tunneling to reach the target, it is possible to use the backup links of a multi-homed site to carry SOS-routed traffic (effectively using tunneling as a source-routing mechanism). Thus, all attack traffic will use the BGP-advertised best route to the target, while traffic from the SOS infrastructure will use the unused available capacity of the target site.

III. Security Analysis

In this section we develop simple analytical models to evaluate the performance of SOS in the face of DoS attacks. In our evaluation, we make certain assumptions: an attacker knows the set of nodes that form the overlay, and can attack these nodes by bombarding them with traffic. However, the attacker does not know the precise functionality (beacons or servlets) of the nodes, nor can it infer them (*e.g.,* by monitoring traffic through the overlay). The bandwidth available to the attacker to launch attack upon the overlay and the target has an upper bound. Furthermore, we assume that the attackers have not breached the security protocols of the overlay, *i.e.,* their packets can always be identified by SOS as being illegitimate. Finally, each legitimate user can access the overlay through a limited number of SOAPs, but different users access the overlay through different SOAPs.

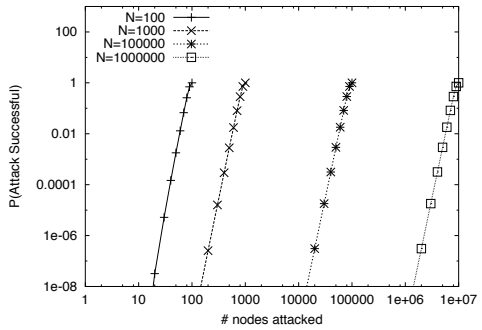
A. A Static Attack

Our analysis begins by considering the following problem: suppose some subset of nodes in the overlay are assigned 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 SOAPs that can be used by a given 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 headed toward T: $U_b = |\{B_i(T)\}|$ is a function of the number of hash functions issued by T.

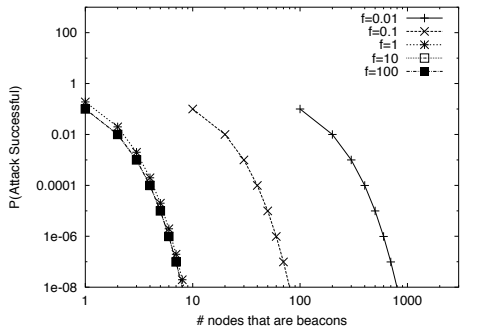
For our initial analysis, we assume that $\mathcal{S}$ can communicate successfully with T as long as there exists an available access point, an available beacon, and an available secret servlet that can be used to complete the communication path. We also assume that the selection of nodes to perform various duties is done independently, such that a node can simultaneously act as any combination of access point, beacon, and secret servlet. We assume that all nodes implement the Chord routing service (and hence can be part of the communication path).

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 nodes. It is easy to show that $P_h(a, b, c) = \binom{b}{c} / \binom{a}{c}$ when $b > c$,[1] and $P_h(a, b, c) = 0$ when $c > b$. 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.

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



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



(b) Varying number of beacons and secret servlets

Fig. 3. Attack success probability for the Static case.

Figure 3 plots the likelihood of an attack succeeding at shutting down access to a site in the static case. In Figure 3(a) we hold U_s, U_b, and U_o 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

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

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 3(b), we hold N fixed at 10^4 and n_a fixed at 10^3. 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 observe 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 3(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. Similarly, Figure 3(b) shows 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 as we increase their numbers the attack success probability rapidly falls beneath minuscule levels. In summary, long-term static attacks upon a moderately-provisioned SOS are unlikely. (Notice that the number of overlay nodes is not limited by the number of POPs; such nodes can be located anywhere throughout the network, even at customer's facilities. If co-located with routers, more than one such node can be attached to each router.)

B. Dynamic Attacks and Recovery

Our previous model assumed that an attacker would select a set of nodes to attack, and that SOS takes no repairing action (*e.g.,* by changing the node that acts as the secret servlet, or by having nodes dropping from the overlay). We extend this model to the case where SOS does take such action and the attacker reacts to a repaired network by altering its attack.

As in the static case, we assume that the attacker has enough bandwidth resources to bring down n_a nodes. When SOS identifies an attacked node, that node is removed from the overlay such that its being attacked does not prevent communication between the source and target. The attacker reacts after some time, and it redirects its attack toward a node that still resides in the overlay. We assume that there is a repair delay, D_r, that equals the difference in time from when a node is first attacked until the time when SOS detects the attack and removes the node. Also, there is an attack delay, D_a, that equals the difference in time between when an attacked node is removed from the overlay to the time when the attacker redirects the attack toward a new node in the overlay.

Our analysis assumes that when an attack on a node is terminated, that node is immediately brought back into the overlay. This is a reasonable assumption since a node can detect when it is no longer being bombarded with traffic.

We define a random variable $A(t)$ to be the number of nodes that are under attack that have not yet been removed from the overlay at time t. Since the attacker can attack up to n_a nodes, we have that $0 \leq A(t) \leq n_a$. Letting $\pi_i = \Pr[A(t) = i]$, we can extend our static case analysis to this dynamic case. Let $U_{\mathcal{S},T}(t)$ be a random variable that equals 1 if $\mathcal{S}$ can reach T during an ongoing attack at time t and 0 otherwise. When i of the n_a nodes are active in the overlay, then the total number of nodes that are active in the overlay is $N + i - n_a$. Then, we obtain:

$$\Pr[U_{\mathcal{S},T}(t) = 1] = \sum_{i=0}^{n_a} \pi_i (1 - P_h(N + i - n_a, i, U_s)) \cdot (1 - P_h(N + i - n_a, i, U_b)) \cdot (1 - P_h(N + i - n_a, i, U_o))$$

where $P_h(a, b, c)$ is set to equal $P_h(a, b, a)$ when $c > a$.

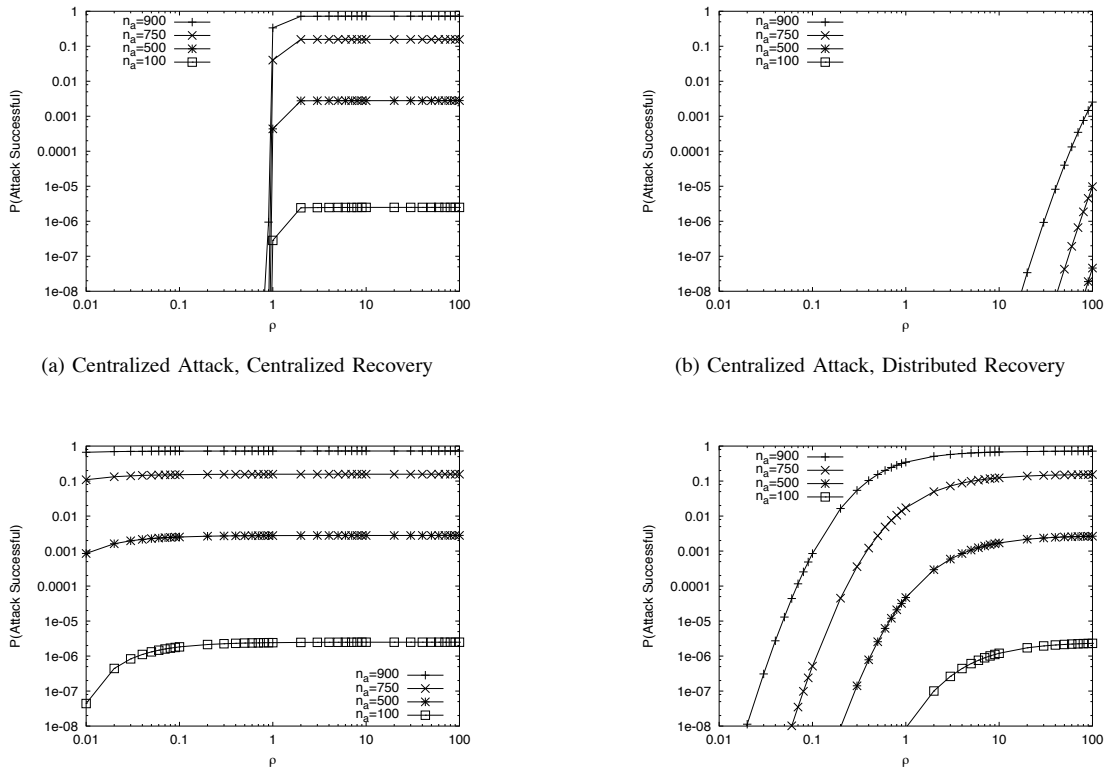
We are interested in two variants of how we model the SOS repair process. In the first, the ability to react to each attacked node is performed sequentially. This would occur when the decision to modify the overlay is made by a single centralized authority. We refer to this variant as the *centralized repair process*. Alternatively, there can be a *distributed repair process*, where repairs can be performed in parallel. This would occur when each node can independently perform its repair process. Similarly, the attack process can be centralized, where only one attack node can be modified at a time, or distributed, where separate attackers are responsible for the detection and movement of their individual attacks.

Because SOS is a novel architecture, we do not yet have a detailed understanding of how the repair and attack processes will function. Thus, we do not have models that accurately capture the distributions of D_a (attack delay) and D_r (repair delay). Nonetheless, we are interested in gaining preliminary insight into how the relative rate of change in the number of successfully attacked nodes active in the overlay affects the robustness of SOS. We achieve this insight by modeling the framework as a closed queuing system with a finite customer population. Customers arrive at the server(s), obtain service and then after a certain delay or think time, return to get serviced again. In these models, the number of jobs active in the queuing system equals the number of nodes actively under attack that remain in the overlay. The repair process removes jobs (service) from the system and the discovery by attacker and redirection places jobs back in the system. We assume both D_a and D_r are exponentially distributed random variables with respective rates λ and μ.

TABLE I
QUEUING MODELS FOR THE VARIANTS OF ATTACK AND REPAIR PROCESSES.

Attack process	Repair process	
	centr.	distr.
centr.	$M/M/1/K$	$M/M/K/K$
distr.	$M/M/1//K$	$M/M///K$

Table I presents the queuing models used to capture the four possible scenarios, given that both the attack and repair processes can be either centralized and distributed. Each of the four models is a birth-death process with $K = n_a + 1$ states, where the process resides in state i when there are i nodes that are active in the overlay that are being attacked, $0 \leq i \leq n_a$. When the attack is centralized, the rate of transition from state i to state $i+1$ is λ. In the distributed case, the rate is $(n_a - i)\lambda$. When the repair is centralized, the rate of transition from state



(a) Centralized Attack, Centralized Recovery

(b) Centralized Attack, Distributed Recovery

(c) Distributed Attack, Centralized Recovery

(d) Distributed Attack, Distributed Recovery

Fig. 4. Attack success probability for the Dynamic case.

i to state $i-1$ is μ. In the distributed case, the rate is $i\mu$.

In each model, π_i is expressed as a function of $\rho = \lambda/\mu$. See [14] for the exact formulas. In Figure 4, we plot $\Pr[U_{\mathcal{S},T}(t) = 1]$, varying $\rho = \lambda/\mu$ along the x-axis. In each figure, the SOS overlay contains 10^3 nodes, where 10 nodes are selected as secret servlets, 10 nodes selected as beacons, and each user can access the overlay through 10 SOAPs. Each curve plots $\Pr[U_{\mathcal{S},T}(t) = 1]$ using a different value for n_a. We see that as ρ grows large, $\Pr[U_{\mathcal{S},T}(t) = 1]$ grows asymptotically to the corresponding value of the static case, $\Pr[U_{\mathcal{S},T} = 1]$. As ρ increases, attacks recover more quickly and repair takes longer, such that the expected number of attacked nodes inside the overlay approaches n_a. When ρ is small, $\Pr[U_{\mathcal{S},T}(t) = 1]$ diminishes since the number of nodes successfully attacked inside the system is reduced.

Not surprisingly, for a fixed ρ, attacks are most likely to deny service to the target when the attack process is distributed and the repair process centralized, and are least likely to deny service when the reverse holds. When both processes are distributed, the fraction of time for which the attack is successful can be significant when a large fraction of nodes in the overlay is attacked, even when $\rho < 1$. This can be understood intuitively by comparing the respective birth-death processes of the system when repair and attack processes are both centralized and where they are both distributed. When both processes are centralized, each upward transition's rate equals λ and each downward transition's rate equals μ. In the system where both processes are distributed, the upward transitions' rates of $(n_a - i)\lambda$ are larger for states with smaller i, whereas the downward transitions' rates of $i\mu$ are smaller with smaller i. As a result, when $\rho < 1$, the centralized-process system is less likely to drift away from the smaller states.

C. Attacking the Underlying Network

To this point, we have assumed that to deny service to a target protected by SOS, an attacker will deny service to nodes in the overlay. Another alternative, however, is to launch an attack at the core of the network. Rather than attacking the edge nodes that make up the overlay, attackers can focus on those core nodes that lie on paths between multiple overlays.

We measure attack severity in a scenario in which several compromised zombie nodes, widely distributed over the network, launch attacks on a target node. The attacks can be coordinated, timer-driven or triggered by events like opening of mailboxes, booting up of zombie machines, *etc*. For instance, the triggering mechanism of the attack can either be (i) attack immediately, or (ii) execute code at some specified time. For (i), the timing of the attack depends on the infection vector: for an email-based worm it is reasonable to assume that attacks will go off at random times from zombie machines. For (ii), we can assume the coordinated attacks to be a single "large" attack. We next show that attacks that are a combination of the two will overpower routers with low bandwidth capabilities much easier than those with high bandwidth capabilities.

As a simple first approximation, we can view the arrival of the attacks from such clients (with coordinated attacks acting as a single, "large" attack client) as a Poisson process, with an arrival rate λ_a attacks per unit time. Note that we are modeling the *attack arrival* as a Poisson process. The attack traffic itself is assumed to be (high bandwidth) CBR. 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 can 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 resources get tied up and arriving requests (legitimate or not) are denied service, we consider the attack successful.

The system model is now abstracted into a Stochastic Knapsack [15] 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. The probability of a successful DoS attack is the blocking probability corresponding to the class of legitimate traffic.

The blocking probability B_m for a class-m call under Poisson arrival assumption is [15]:

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

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

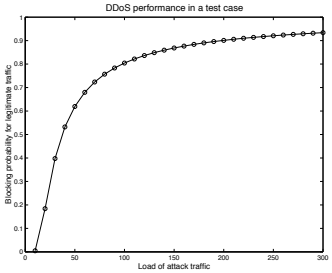
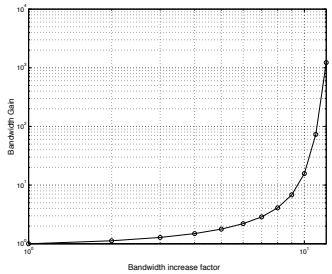


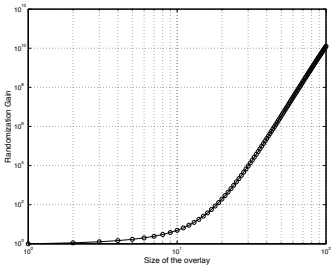
Fig. 5. Blocking probability for legitimate traffi c as a function of attack traffi c load.

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. 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*., however that does not change the nature of the results we present. 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 $\rho = \lambda/\mu$ for the legitimate traffic is 1. In Figure 5, 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 10^4, 99.8% of legitimate traffic is denied service. Now we consider the effects of two key features of the SOS architecture. 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 10Gbps 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 6(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.

Fig. 6. Performance gains with SOS.

Next, we study the effects of anonymizing the attacked node. If the attacker does not know the identity of the secret servlet for a particular target, the attacks will be launched randomly onto the overlay. Only a fraction of those attacks

will reach the target servlet. Thus, the effective arrival rate of the attacks becomes $\lambda_a \times f$, 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 6(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 increases from left to right, a correspondingly 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.

IV. Performance Measurements

In order to quantify the overhead associated with use of SOS, we developed a prototype implementation using web proxies as the SOS overlay nodes, and measured the time-to-completion of https requests. That is, we measured the elapsed time starting when the browser initiates the TCP connection to the destination or the first proxy, to the time all data from the remote web server have been received. We ran this test by contacting 3 different SSL-enabled sites: *login.yahoo.com, www.verisign.com,* and the Columbia course bulletin board web service (`https://www1.columbia.edu/sec/bboard`). For each of these sites, we measured the time-to-completion for a different number of overlay nodes between the browser and the target (remote web server).

Table II shows the results for the case of 0 (browser contacts remote server directly), 1, 4, 7, and 10 overlay nodes. The times reported are in seconds, and are averaged over several HTTPS GET requests of the same page, which was not locally cached. For each GET request, a new TCP connection was initiated by the browser. The row labeled "Columbia (2nd)" shows the time-to-completion of an HTTPS GET request that uses an already-established connection through the overlay to the web server, using the HTTP 1.1 protocol.

TABLE II
DIRECT CONNECTION VERSUS CHORD-BASED SOS OVERLAY.

Server	Direct	1 node	4 nodes	7 nodes	10 nodes
Yahoo!	1.39	3.15	5.53	10.65	14.36
Verisign	3.43	5.12	7.95	14.95	22.82
Columbia BB	0.64	1.01	1.45	3.14	5.07
Columbia (2nd)	0.14	0.23	0.28	0.57	0.72

We used PlanetLab [16], a wide-area overlay network testbed. The PlanetLab nodes are distributed in academic institutions across the country, connected over the Internet. We deployed our SOS proxies PlanetLab and ran the exact same tests. We see that the time-to-completion in this scenario increases by a factor of 2 to 10, depending on the number of nodes in the overlay. The increase in latency can be directly attributed to the delay in the links between the SOS nodes.

While the PlanetLab configuration allowed us to conduct a much more realistic performance evaluation, it also represents a worst-case deployment scenario for SOS: typically, we would expect SOS to be offered as a service by an ISP, with the (majority of) SOS nodes located near the core of the network. Using PlanetLab, the nodes are distributed in (admittedly well-connected) end-sites. We would expect that a more commercial-oriented deployment of SOS would result in a corresponding decrease in the inter-overlay delay. On the other hand, it is easier to envision end-site deployment of SOS, since it does not require any participation from the ISPs. Finally, while the additional overhead imposed by SOS can be significant, we have to consider the alternative: no web service while a DoS attack against the server is occurring. While an increase in end-to-end latency by a factor of 5–10 is considerable, we believe it is more than acceptable in certain environments and in the presence of a determined attack.

Table III shows the results (in seconds) when a shortcut implementation was tested on the PlanetLab testbed, using 76 overlay nodes. In this variant, SOAPs use Chord routing to contact the beacon and determine the identity of the appropriate secret servlet. They then cache this information for use with subsequent traffic between the source and the target, and directly route traffic to the servlet. Thus, in this scenario, the overlay itself is used only for signaling, with actual data transfer requiring only two hops. The hops to the beacon ranged from 4 to 8 and did not have a significant effect on latency. This implementation provides significant performance improvements, particularly on subsequent requests for the same site because of the caching, with end-to-end latency increasing by as little as a factor of 2. To simulate the effects of an attack on individual nodes in the overlay, we brought down specific nodes. The overlay healed itself within 10 seconds.

TABLE III
DIRECT CONNECTION VERSUS SHORTCUT-BASED SOS OVERLAY.

Server	Direct	Original Request	Cached Requests
Yahoo!	1.39	4.15	3.67
Verisign	3.43	7.33	6.77
Columbia BB	0.64	3.97	3.43
Columbia (2nd)	0.14	0.55	0.56

V. Further Discussion

Our study of SOS is admittedly in its early stages. There are several issues that need to be addressed 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 user 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. Users of the infrastructure should treat it as an untrusted network in terms of privacy or integrity (*i.e.,* if their communications are of a sensitive nature, they should be

appropriately encrypted) — SOS only attempts to address the DoS problem; as such, it should be treated as a virtual WAN.

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. This would 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 networks.

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. Preliminary simulations have shown the latency to be in the order of 10 times larger than in the direct communications case (in the absence of an attack). While this is a large overhead, it may be acceptable in mission-critical systems. 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.

VI. Related Work

A fundamental design principle of the IP architecture is to keep the functionality inside the core of the network simple, pushing as much mechanism as possible to the network end-points. This principle, commonly referred to as the "end-to-end principle"[17], has been the basic premise behind protocol design. However, as has been demonstrated in the past few years [18], [19], [20], such mechanisms are inadequate in addressing the problem of DoS attacks.

Unfortunately, as a result of its increased popularity and usefulness, the Internet contains both interesting targets and enough malicious and 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 [21].

The need to protect against or mitigate the effects of DoS attacks has been recognized by both the commercial and research world. Some work has been done toward achieving these goals, *e.g.,* [9], [22], [23], [24]. However, 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 sources) as possible. Thus, they fall into this class of approaches that are reactive. The motivation behind such approaches 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 and used by end-systems. Second, these mechanisms are fairly transparent to protocols, applications, and legitimate users. Unfortunately, these reactive approaches by themselves are not always adequate solutions.

Methods that filter traffic by looking for known attack patterns or statistical anomalies in traffic patterns can be defeated by changing the attack pattern and masking the anomalies that are sought by the filter. Furthermore, statistical approaches will likely filter out valid traffic as well. 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 (often taking several 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 just start using another compromised node.

Using a "pushback"-like mechanism such as that described in [9] 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 [25]), 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 risk of voluntarily cutting off or adversely impacting (*e.g.,* by rate-limiting) its communications with the rest of the Internet. The accuracy of such filtering mechanisms improves dramatically as the filters are "pushed" closer to the actual source(s) of the attack. Thus, it will be necessary for providers to allow other providers, or even end-network administrators, to install filters on their routers. Apart from the very realistic possibility of abuse, it is questionable whether such collaboration can be achieved to the degree necessary.

The same concerns hold for the case of collaborative action by the ISPs: even easy to implement mechanisms such as ingress filtering, that could reduce or even eliminate spoofed-address DoS attacks, are still not in wide use. We believe it is rather unrealistic to expect that cooperative providers would even establish static filters to allow legitimate (paying) clients to tunnel through their infrastructure with any assurance of quality of service, and much less so for the case of mobile or remote clients (as may be the case for emergency teams).

Another approach to mitigating DoS attacks against information carriers is to massively replicate the content being secured around the entire network. To prevent access to the replicated information, an attacker must attack all replication points throughout the entire network — a task that is considerably more difficult than attacking a small number of, often co-located, servers. Replication is a promising means to preserve information that is relatively static, such as news articles. However, there are several reasons why replication is not always an ideal solution. For instance, the information may require frequent updates, complicating large-scale coherency (especially during DoS attacks), or may be dynamic by its very nature (*e.g.,* live audio or video). Another concern is the security of the information: engineering a highly-replicated

solution without information leaks is a challenging endeavor.

VII. 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*.

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.

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

[1] A. D. Keromytis, V. Misra, and D. Rubenstein, "SOS: Secure Overlay Services," in *Proceedings of ACM SIGCOMM*, August 2002, pp. 61–72.

[2] M. Blaze, J. Ioannidis, and A. Keromytis, "Trust Management for IPsec," in *Proceedings of Network and Distributed System Security Symposium*, February 2001, pp. 139–151.

[3] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, "Resilient Overlay Networks," in *Proceedings of the 18th Symposium on Operating Systems Principles (SOSP)*, October 2001.

[4] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications," in *Proceedings of ACM SIGCOMM*, August 2001.

[5] F. Dabek, M. F. Kaashoek, R. Morris, D. Karger, and I. Stoica, "Wide-Area Cooperative Storage with CFS," in *Proceedings of ACM SOSP*, October 2001.

[6] M. G. Reed, P. F. Syverson, and D. M. Goldschlag, "Anonymous connections and onion routing," *IEEE Journal on Special Areas in Communications*, vol. 16, no. 4, pp. 482–494, 1998.

[7] M. Wright, M. Adler, B. N. Levine, and C. Shields, "An Analysis of the Degradation of Anonymous Protocols," in *Proceedings of the ISOC Symposium on Network and Distributed System Security*, February 2001.

[8] S. Ioannidis, A. Keromytis, S. Bellovin, and J. Smith, "Implementing a Distributed Firewall," in *Proceedings of Computer and Communications Security (CCS)*, November 2000, pp. 190–199.

[9] J. Ioannidis and S. M. Bellovin, "Implementing Pushback: Router-Based Defense Against DDoS Attacks," in *Proceedings of the Network and Distributed System Security Symposium*, February 2002.

[10] M. K. Reiter and A. D. Rubin, "Crowds: anonymity for Web transactions," *ACM Transactions on Information and System Security*, vol. 1, no. 1, pp. 66–92, 1998.

[11] D. Karger, E. Lehman, F. Leighton, R. Panigrahy, M. Levine, and D. Lewin, "Consistent Hashing and Random Trees: Distributed Caching Protocols for Relievig Hot Spots on the World Wide Web," in *Proceedings of ACM Symposium on Theory of Computing (STOC)*, May 1997, pp. 654–663.

[12] E. Sit and R. Morris, "Security Considerations for Peer-to-Peer Distributed Hash Tables," in *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS)*, March 2002.

[13] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Content-Addressable Network," in *Proceedings of ACM SIGCOMM*, August 2001.

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

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

[16] L. Peterson, D. Culler, T. Anderson, and T. Roscoe, "A Blueprint for Introducing Disruptive Technology into the Internet," in *Proceedings of the 1st Workshop on Hot Topics in Networks (HotNets-I)*, October 2002. [Online]. Available: citeseer.nj.nec.com/peterson02blueprint.html

[17] J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-to-end arguments in System Design," *ACM Transactions on Computer Systems*, vol. 2, no. 4, pp. 277–288, November 1984.

[18] C. Schuba, I. Krsul, M. Kuhn, E. Spafford, A. Sundaram, and D. Zamboni, "Analysis of a Denial of Service Attack on TCP," in *Proceedings of IEEE Security and Privacy*, May 1997, pp. 208–223.

[19] L. Heberlein and M. Bishop, "Attack Class: Address Spoofi ng," in *Proceedings of the 19th National Information Systems Security Conference*, October 1996, pp. 371–377.

[20] S. Savage, N. Cardwell, D. Wetherall, and T. Anderson, "TCP Congestion Control with a Misbehaving Receiver," *ACM Computer Communications Review*, vol. 29, no. 5, pp. 71–78, October 1999.

[21] D. Moore, G. Voelker, and S. Savage, "Inferring Internet Denial-of-Service Activity," in *Proceedings of the 10th USENIX Security Symposium*, August 2001, pp. 9–22.

[22] D. Dean, M. Franklin, and A. Stubblefi eld, "An Algebraic Approach to IP Traceback," in *Proceedings of the Network and Dsitributed System Security Symposium*, February 2001, pp. 3–12.

[23] S. Savage, D. Wetherall, A. Karlin, and T. Anderson, "Network Support for IP Traceback," *ACM/IEEE Transactions on Networking*, vol. 9, no. 3, pp. 226–237, June 2001.

[24] M. T. Goodrich, "Effi cient Packet Marking forLArg-Scale IP Traceback," in *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS)*, November 2002, pp. 117–126.

[25] K. Park and H. Lee, "On the Effectiveness of Route-Based Packet Filtering for Distribtued DoS Attack Prevention in Power-Law Internets," in *Proceedings of ACM SIGCOMM*, August 2001, pp. 15–26.

Angelos D. Keromytis (M'97) received his Ph.D. in Computer Science from the University in Pennsylvania, Philadelphia, PA, in 2001. He received his B.S. in Computer Science from the University of Crete, Heraclion, Greece, in 1996. Currently he is an Assistant Professor of Computer Science at Columbia University, New York, since July 2001. His research interests include design and analysis of network and cryptographic protocols, software security and reliability, and operating system design.

Vishal Misra (M'97) is an Assistant Professor in the Departments of Computer Science and Electrical Engineering at Columbia University since 2001. He received his BTech from IIT Bombay, and M.S. and Ph.D. from the University of Massachusetts, Amherst, all in Electrical Engineering. His research interests are in the modeling, analysis and design of algorithms for communication networks. He is a recipient of CAREER awards from NSF and DoE.

Dan Rubenstein (M'97) has been an Assistant Professor of Electrical Engineering and Computer Science at Columbia University since 2000. He received a B.S. degree in mathematics from M.I.T., an M.A. in mathematics from UCLA, and a Ph.D. in computer science from the University of Massachusetts, Amherst. His research interests are in network technologies, applications, and performance analysis, with an emphasis on large-scale Internet design for continuous media transmission. He received a Best Student Paper Award for his ACM SIGMETRICS 2000 paper entitled "Detecting Shared Points of Congestion via End-to-end Measurement" and an NSF CAREER Award in 2002 to continue his investigation of peer-to-peer and overlay networking systems.