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

ABSTRACT

Denial of service (DoS) attacks continue to threaten the reliability of networking systems. Previous approaches for protecting 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 toward supporting Emergency Services or similar types of communication. The architecture is constructed using 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.

Categories and Subject Descriptors

C.2.0 [**Security and Protection**]: Denial of Service; C.2.1 [**Network Topology**]: Overlay Networks

General Terms

Security, Reliability.

*This material 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. 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.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SIGCOMM'02, August 19-23, 2002, Pittsburgh, Pennsylvania, USA.
Copyright 2002 ACM 1-58113-570-X/02/0008 ...$5.00.

Keywords

Network Security, Denial of Service Attacks, Overlay Networks.

1. INTRODUCTION

In the immediate aftermath of 9/11 events in New York City, the Internet was used to facilitate communication between family members and friends, as the phone network was overwhelmed[1]. It does not require a great leap of faith to imagine using the Internet as a communication medium for crisis and emergency response teams. In particular, the network could be used to carry communications between widely dispersed "static" sites (*e.g.,* various federal, state, and city agencies) and (semi-) roaming stations and users. In such an environment, the communication path between the various sites and the emergency response teams (ERTs) needs to be kept clear of interference such as denial of service (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 (them) with bogus packets. Such attacks on a network could seriously disrupt the rescue and recovery effort at minimal cost and danger to the attacker. Yet the Internet, both in its current form and in the form that it will likely evolve into the future, will remain a fundamentally open network. Thus, we cannot reasonably expect any one entity to *effectively* police and control it.

A proposal to build a completely separate network is under consideration by the US government (appropriately named "GovNet"). Such a network would have a high deployment and maintenance 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.

In this paper, we address the problem of securing communication on top of today's existing IP infrastructure from DoS attacks, where the communication is between a pre-determined location and 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 subset of clients scattered throughout the wide-area network who require (and should have) access to this information. These users

[1]In addition to the well-documented increase in email traffic, the Computer Science Department at Columbia University set up Voice over IP gateways that worked flawlessly when the telephone network came to a standstill.

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 whose jamming will 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.

We should stress that our approach does not solve the general DoS problem (*e.g.,* the problem of distinguishing between good- and bad-intentioned requests to a web server such as Google). We are interested in classes of communication where both participants are known to each other. While our work was motivated by the ERT scenario, it is equally applicable to any other environment where both parties have some form of pre-established trust relationship (*e.g.,* telecommuting, corporate Intranets, subscribers to a news website, *etc.*).

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) so that the effects of the attack are minimized.

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.

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 overlay such that it is secure with high probability, given attackers who have a large but finite set of resources to perform the attacks. The attackers 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 also assume that the attacker does not have unobstructed access to the network core. That is, our model allows for the attacker to take over an arbitrary (but finite) number of hosts, but only a small number of routers. In particular, the attacker can bypass our mechanism if they can take control of a router that lies in the path between one of the "approved" overlay nodes and the target's filtering router: at that point, the attacker knows an IP address that will be allowed to reach the target and can use that to launch the DoS attack.

We feel that this assumption is justified since, if the attacker can gain control of such a router, they can cut off the target's communications directly, without the need to launch a DoS attack. Also, it is more difficult (but not impossible) to take control of a router than an end-host or server, due to the limited number of potentially exploitable services offered by the former. While routers offer very attractive targets to hackers, there have been very few confirmed cases where take-over attacks have been successful.

Furthermore, we assume that the attacker cannot acquire sufficient resources to severely disrupt large portions of the backbone itself (*i.e.,* such that all paths to the target are congested).

Our architecture leverages heavily off of previous work on IP security[4], IP router filtering capabilities, and novel approaches to routing in overlays[1] and peer-to-peer (P2P) networks[26, 6]. 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 [20] 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 who can communicate access the overlay through a small subset of overlay nodes.

The remainder of the paper proceeds as follows. We review related work in Section 2. Section 3 describes the SOS Architecture, whose resistance to attacks is evaluated in Section 4. Section 5 discusses implementation details, Section 6 provides a general discussion of our results and future directions, and Section 7 concludes the paper.

2. 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"[22, 5], has been the basic premise behind protocol design. However, as has been demonstrated in the past few years [25, 10], such mechanisms are inadequate in addressing the problem of DoS attacks.

It is trivial to abuse[23] 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 to launch one of these attacks. Even relatively large-scale DoS attacks (Distributed DoS — DDoS)[2] 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.

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

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 [18]. Worse, the Internet is increasingly being used for 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 research world. Some work has been done toward achieving these goals, *e.g.,* [12, 7, 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 [12] 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 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.,* a live audio or video stream). Another concern is the security of the stored information: engineering a highly-replicated solution without "leaks" of information is a challenging endeavor.

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 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 shall discuss in Section 5 how this can be efficiently achieved for a large collection of SOS nodes and users. 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, such as ERT personnel, 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 SOS is known to the public and hence is also known 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 by the architecture.

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 highly specialized and sophisticated attackers is beyond the scope of this paper.

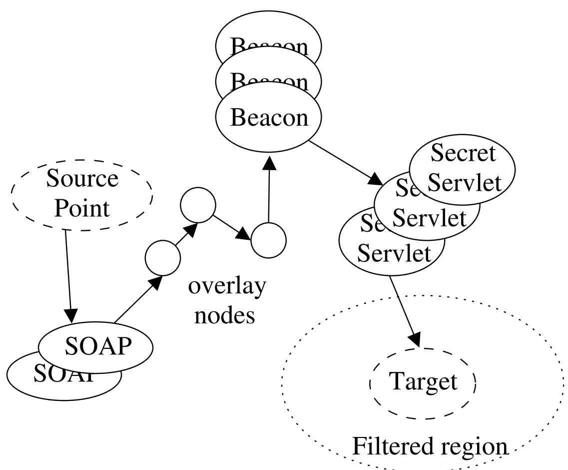


Figure 1: Basic SOS architecture.

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.

3.1 Design Rationale

Fundamentally, the goal of the SOS infrastructure is to distinguish between authorized and unauthorized traffic. The former is allowed to reach the destination, while the latter (or, more generally, unverified traffic) is dropped or is rate-limited. Thus, at a very basic level, we need the functionality of a firewall "deep" enough in the network that the access link to the target is not congested. This imaginary firewall would perform access control by using traditional 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 [13]. 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 ("pinch points") 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 upon 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, however, 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.

3.2 Protecting the Target: Filtering

In the current Internet, knowledge of the network identifier (IP address) of the target 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. Typically, such filters need to be established at the ISP's Point of Presence (POP) routers that attach to the ISP backbone.

3.3 Reaching a 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 addresses, or ceases to be legitimate, the filter surrounding the target must be notified and 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 known legitimate user.

A first step in our solution is have the target select a subset of nodes, N_s, that participate in the SOS overlay to act as *forwarding proxies*. The filter is set to only allow 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. Thus, it can use more sophisticated (and heavy-weight) security 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) secret servlets. All other traffic can be dropped, or rate-limited. Because of the small number of such filter rules

(3-4 per interface) and their simple nature (filter only on source IP address), router performance will not be impaired[12], even if we do utilized specialized hardware (*e.g.,* CAMs).

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*.

3.4 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 it 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 destination (a secret servlet) while utilizing a minimal amount of information about the identity of that destination.

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 simply removing those "shut down" nodes from the overlay and routing around them.

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. This is to have each overlay node that receives a packet randomly choose the next hop on the overlay to which it forwards a packet. Eventually, the packet will arrive at a secret servlet, which can then deliver the packet to the target.

3.5 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. For this purpose, off-the-shelf authentication protocols such as IPsec or TLS can be used. 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 [13] 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 effectively deny connectivity to legitimate clients.

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. An investigation into how SOAPs are chosen is part of our future work plans.

3.6 Routing through the Overlay

Having each overlay participant select the next node at random is sufficient to eventually reach a secret servlet. 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 a possible 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 $O(\log N)$.

The routing algorithm utilizes the Chord service [26]. For the purposes of this paper, Chord can be viewed as a routing service that can be implemented atop the existing IP network structure (*i.e.,* in a network overlay). Consistent hashing [14] is used to map, by using a hash function, an arbitrary identifier to a unique destination node that is an active member of the overlay. Each overlay node maintains a list that contains $O(\log N)$ identities of other active nodes in the overlay.

Given the destination identifier, each node knows how to choose a member in its list such that, from an arbitrarily chosen starting node, the destination node to which the identifier hashes is reached in $O(\log N)$ overlay hops. Multiple destination nodes for a given identifier can be created by using different hash functions. In addition, by choosing the right class of hash functions, the sequences of nodes used to carry a packet from a node to the destination are independent from one another (uncorrelated). It is simple to produce multiple mappings (hash functions) that produce different paths to different sets of destination nodes (*i.e.,* the sets of nodes that form paths from a given start node to a given sink node are independent).

The Chord service is robust to changes in overlay membership: each node's list is adjusted to account for nodes leaving and joining the overlay such that the above stated properties continue to hold.

In SOS, the identifier used to which the hash function is applied is the IP address of the target. Thus, Chord can be used to direct a packet 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 (after possibly several overlay hops) using Chord, regardless of the starting point in the overlay. We refer to this node as the *beacon*, since it is to this node that packets destined for the target are first guided. When a packet is approved by a SOAP for forwarding over the overlay, the hash on the IP address of the target is used as the key. Chord therefore provides a robust and reliable while relatively unpredictable means of routing packets from an access point to one of several beacons.

One last step needed is to provide the mechanism that communicates the secret servlet's identity to the beacon node. This can also be achieved via Chord: as an overlay node, the secret servlet or the target can reach the beacon by hashing on the target identifier (which the secret servlet knows), and then using Chord routing in the same way that SOAPs do. Hence, the secret servlet or the target can inform the beacon of the secret servlet's identity.

By providing only the beacon with the identity of the secret servlet, the packet can be delivered from any SOAP 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 filter) to the target. This allows the overlay to scale for arbitrarily large numbers of overlay nodes and target sites. If the overlay only serves a relatively small number of target sites, traditional routing protocols or RON-like routing[1] may be sufficient.

3.7 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, as discussed above, 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, and the legitimate user attempting access need only find another SOAP that will accept its transmissions.

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 ID 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 via 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 multihomed 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.

3.8 Summary of Architecture

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

1. 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 do filtering (using only a small number of rules) on incoming traffic without adversely impacting their performance.

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. 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 or the target will contact them, notifying the beacons of the servlets' existence. Beacons, after verifying the validity of the request, will store the necessary information to forward traffic for that target to the appropriate servlet.

4. 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.

5. The beacon routes the packet to a secret servlet that then routes the packet (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.

- 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.

4. PERFORMANCE ANALYSIS OF SOS

In this section we develop simple analytical models that describe DoS attacks, and evaluate the SOS architecture using these models to evaluate its resilience to DoS attacks. Our evaluation makes the following assumptions:

- An attacker knows the set of nodes that form the overlay, and can attack these nodes by bombarding them with traffic. There is a fixed amount of bandwidth the attacker can use to mount its attack upon the overlay and the target.

- An attacker does not know which nodes are secret servlets or beacons, and does not infer these identities (*e.g.,* by monitoring traffic through the overlay).

- Attackers have not breached the security protocols of the overlay, *i.e.,* their packets can always be identified by SOS nodes as being illegitimate.

- Each legitimate user can access the overlay through a limited number of SOAPs, but different users access the overlay through different SOAPs. Thus, an attacker that wants to prevent all communication to the target will not target the SOAPs of a specific user, since doing so only ensures that only that user cannot communicate with the target.

Our evaluation will determine the likelihood that an arbitrarily chosen user's communication to the target is prevented by the attack. This likelihood is clearly lower than the likelihood that all communication to the target is successfully prevented, but higher than the likelihood that there exists a user that cannot access the target.

4.1 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. This assumes that all beacons are aware of all secret servlets. The analysis is easily extended to 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 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). Note that Chord will be able to route effectively even if only one node remains in the overlay, though the node will have to simultaneously be the access point, beacon, and secret servlet.

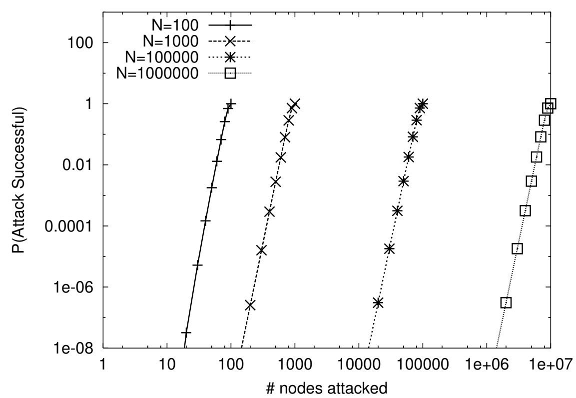
Let $P_h(a, b, c)$ be the probability that a set of b nodes selected at random from $a \geq b$ nodes contains a specific subset of c nodes. It is easy to show that $P_h(a, b, c) = \binom{b}{c} / \binom{a}{c}$ when $b > c$,[3] and $P_h(a, b, c) = 0$ when $c > b$.

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.

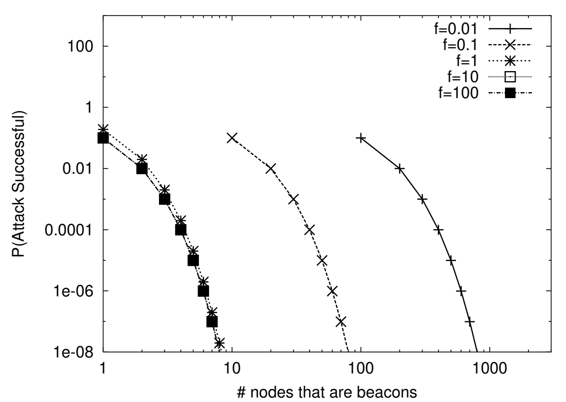
$$\Pr[U_{T,\mathcal{S}} = 1] = (1 - P_h(N, n_a, U_s)) \cdot (1 - P_h(N, n_a, U_b)) \cdot (1 - P_h(N, n_a, U_o))$$

Figure 2 plots the likelihood of an attack succeeding at shutting down access to a site in the static case. In Figure 2(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 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 2(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 $f = U_s/U_b$.

[3] 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 2: Attack success probability for the Static case.

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 2(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 2(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 the numbers needed to force attacks to be successful beneath minuscule probabilities are quite small. 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.)

4.2 Dynamic Attacks and Recovery

Our previous model assumed that an attacker would select a set of nodes to attack, and that SOS takes no action toward repairing the attack (*e.g.,* by changing the node that acts as the secret servlet, or by having nodes from from their participation in the overlay). Here, 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. When an attacker identifies that a node it is attacking no longer resides in the overlay, it redirects its attack toward a node that does still reside 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 from the overlay. 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 (realizing the node it is attacking has been removed) 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. Under this model, the attacker prevents communication during the period of time it takes SOS to identify the attacked node, remove it from the overlay, and repair the overlay to complete communication to the target (*e.g.,* change the node that acts as the secret servlet, allow Chord to reconfigure in response to deletions in the overlay, *etc.*).

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 $0 \le A(t) \le n_a$. 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 $N + i - n_a$. Assuming the attack and recovery system has reached a steady state, we have:

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

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 applying fundamental queueing models to capture the attack and repair behaviors. In these models, the number of jobs active in the queueing system equals the number of nodes actively under attack that remain in the overlay. The repair process removes jobs from the system and the attack process places jobs in the system. We assume both D_a and D_r are exponentially distributed random variables with respective rates λ and μ.

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

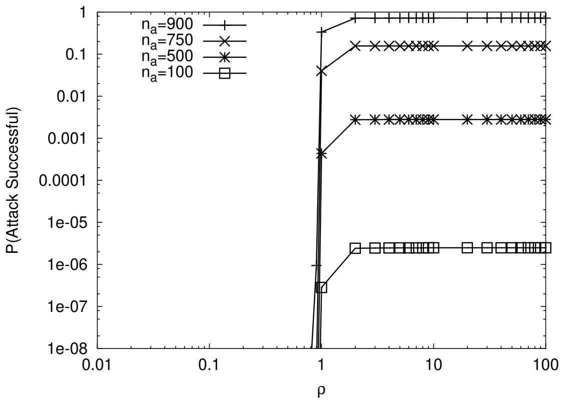
Table 1: Queueing models for the variants of attack and repair processes.

Table 1 presents the queueing 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, $0 \le i \le n_a$. 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 i to state $i - 1$ is μ. In the distributed case, the rate is $i\mu$.

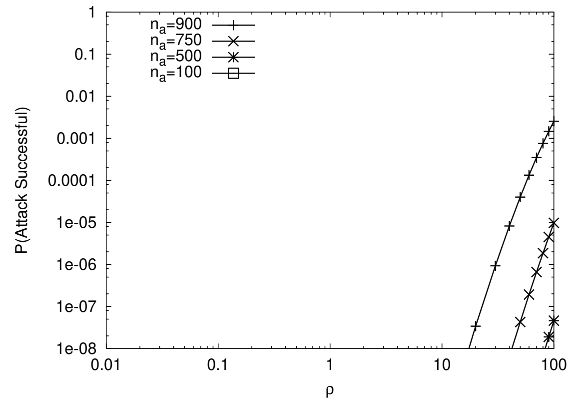
In each model, π_i is expressed as a function of $\rho = \lambda/\mu$. See [17] for the exact formulas.

In Figure 3, 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.

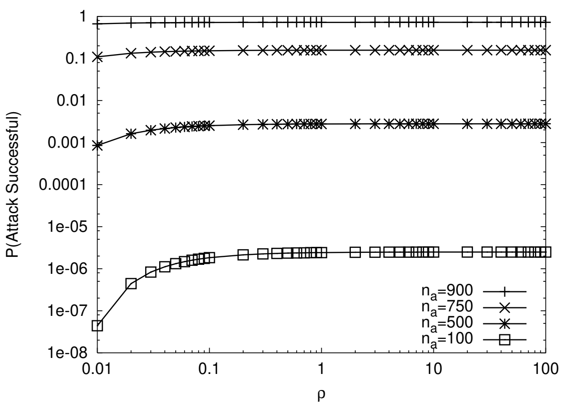
Not surprisingly, for a fixed ρ, attacks are most likely to deny service to the target when the attack process is distributed and the repair process centralized, and are least likely to deny service when the attack process is centralized and the repair process is distributed. One interesting result is that the sensitivity of $\Pr[U_{\mathcal{S},T}(t) = 1]$ to ρ is much higher when both attack and repair processes are centralized than when both processes are distributed. When both processes are centralized, the fraction of time in which an attack is successful is negligible when $\rho < 1$ (the repair process is faster than the attack process). However, as ρ is increased past 1, the fraction increases quickly toward the asymptotic limit. 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. In the system where 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.



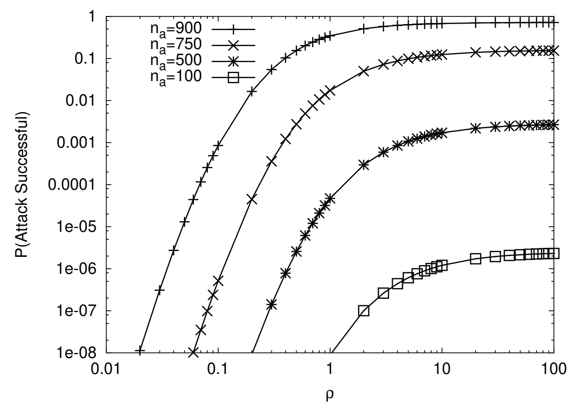
(a) Centralized Attack, Centralized Recovery



(b) Centralized Attack, Distributed Recovery



(c) Distributed Attack, Centralized Recovery



(d) Distributed Attack, Distributed Recovery

Figure 3: Attack success probability for the Dynamic case.

4.3 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[4]. 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 the resources get tied up, arriving requests, legitimate or not, are denied service. We then say that a DoS attack is successful.

The system model is now abstracted into a Stochastic Knapsack [21] 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

[4]Note that we are modeling the *attack arrival* as a Poisson process. The attack traffic itself is assumed to be (high bandwidth) CBR.

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 using the $m-$dimensional integer vector $\mathbf{n}$, *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.,*

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

The blocking probability B_m for a class-m call under Poisson arrival assumption is then given by [21]

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

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

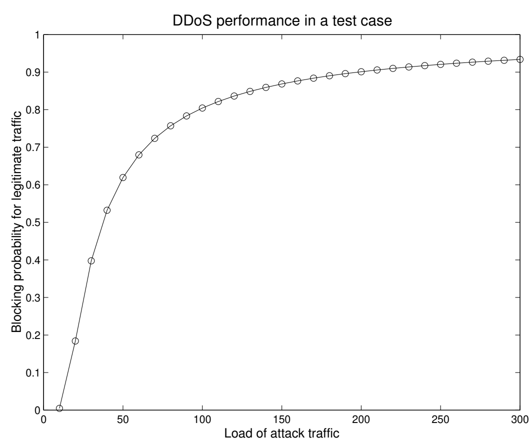
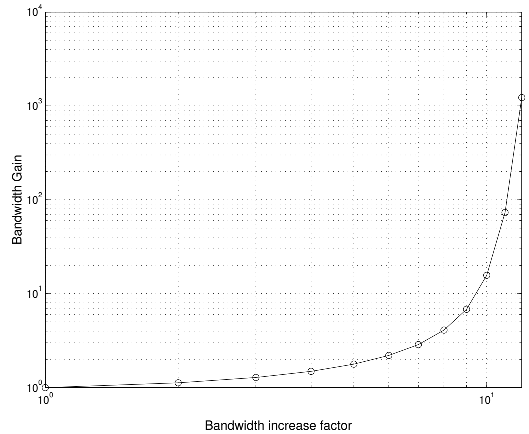


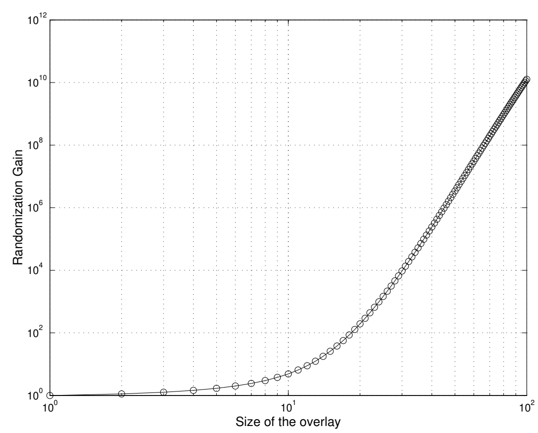
Figure 4: Blocking probability for legitimate traffic as a function of attack traffic load.

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.[5] 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 4, 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 5(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 5: 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

[5] 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.

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 5(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.

5. IMPLEMENTATION OF SOS

One particularly attractive feature of the SOS architecture is that it can be implemented using existing software and standardized protocols, making its adoption and eventual use easier.

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 [12] 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[15]. 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 authenticate) the traffic between them. As described in [4], the conditions under which access to the overlay is allowed can be efficiently encoded in KeyNote credentials[3], which resemble public-key certificates with authorization information embedded. 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, using the techniques presented in [16].

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 key-exchange protocol, named IKE [9]), 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 (by examining the authorization information embedded in the credentials). 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 and certificates.

Tunneling: once traffic has entered the overlay network, it needs to be forwarded to other SOS nodes toward the beacon, and from there to the secret servlets. Standard traffic tunneling techniques[11, 1] and protocols can be used to this end: IP-in-IP encapsulation[19], GRE encapsulation[8], or IPsec in "tunnel mode". Furthermore, traffic inside the overlay network can take advantage of traffic prioritization schemes such as MPLS or DiffServ[2], if they are made available by the infrastructure providers. The routing decisions inside the overlay network are based on a Chord-like mechanism[26].

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 can 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 can also act as "charging" points, if SOS-like functionality is offered on a commercial basis. Finally, since overlay nodes are only called upon to do encapsulated-packet forwarding, cross-provider collaboration[6] is a straightforward proposition, compared to controlled exposure of the filtering mechanism among different providers.

6. 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. Naturally, 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 absense 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.

Analysis: The 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

[6]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.

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. 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.

8. ACKNOWLEDGEMENTS

The authors wish to thank the anonymous reviewers for their valuable comments and suggestions.

9. REFERENCES

[1] 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.

[2] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An Architecture for Differentiated Services. Technical report, IETF RFC 2475, December 1998.

[3] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. D. Keromytis. The KeyNote Trust Management System Version 2. Internet RFC 2704, September 1999.

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

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

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

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

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

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

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

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

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

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

[14] 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)*, pages 654–663, May 1997.

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

[16] A. D. Keromytis. *STRONGMAN: A Scalable Solution To Trust Management In Networks*. PhD thesis, University of Pennsylvania, Philadelphia, 2001.

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

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

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

[20] M. G. Reed, P. F. Syverson, and D. M. Goldschlag. Anonymous connections and onion routing. *IEEE Journal on Special Areas in Communications*, 16(4):482–494, 1998.

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

[22] 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.

[23] 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.

[24] 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.

[25] 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*, pages 208–223, May 1997.

[26] 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*, 2001.