AFRL-IF-RS-TR-2004-236
Final Technical Report
August 2004

SECURE OVERLAY SERVICES (SOS)

Columbia University

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. N523

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

STINFO FINAL REPORT

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2004-236 has been reviewed and is approved for publication

APPROVED: /s/
KEVIN A. KWIAT
Project Engineer

FOR THE DIRECTOR: /s/
WARREN H. DEBANY, Jr.
Technical Advisor, Information Grid Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	Aug 04	Final Jun 02 – Jun 04

4. TITLE AND SUBTITLE

SECURE OVERLAY SERVICES (SOS)

5. FUNDING NUMBERS

C - F30602-02-2-0125
PE - 62301E
PR - N523
TA - FT
WU - N1

6. AUTHOR(S)

Angelos D. Keromytis, Vishal Misra, Dan Rubenstein

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Columbia University Computer Science Dept
1214 Amsterdam Avenue, 515 CS Building
New York, NY 10027

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Defense Advanced Research Projects Agency
3701 North Fairfax Drive
Arlington, VA 22203-1714

AFRL/IFGA
525 Brooks Road
Rome, NY 13441-4505

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

AFRL-IF-RS-TR-2004-236

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: Kevin A. Kwiat, IFGA, 315-330-1692, kwiatk@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 Words)***

The goal of the Secure Overlay Services (SOS) project is to develop an infrastructure upon the existing, insecure Internet that allows an organization to install entities inside the network. An example of an entity is a military database that maintains timely or confidential information (e.g., intelligence). The SOS allows authorized users located anywhere in the Internet to communicate with the entity, and prevents unauthorized users from communicating with the entity. Furthermore, the SOS prevents unauthorized users from denying authorized users access to the entity.

14. SUBJECT TERMS

network overlay, security, Internet, IP

15. NUMBER OF PAGES

21

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Abstract

In the Secure Overlay Services (SOS) project, we investigated the use of overlay network technologies as a means for defeating denial of service (DoS) attacks. Through a use of overlay tunneling, routing via consistent hashing, and filtering at a very small number of routers, we reduce the probability of successful attacks while only increasing the end-to-end latency of communications using SOS by a factor of 2 to 3.

Contrary to most other work in network denial of service, our system provides a means for ensuring the existence of an un-congested end-to-end communications channel between clients and servers in an IP network. SOS does not require changes in existing protocols or infrastructures, and can be deployed in an incremental fashion without collaboration from Internet Service Providers (ISPs). Furthermore, SOS need not be used, and thus need not affect the performance or other characteristics of communications, when no denial of service is taking place in the network.

Our conclusions are that incrementally deployed, overlay-based mechanisms can be very effective in mitigating the impact of denial of service attacks in certain environments, without requiring infrastructure or protocol changes.

The project web page may be found at
http://nsl.cs.columbia.edu/projects/sos/

This work is also currently supported by Cisco and Intel Corp.

Table of Contents

List of Figures and Tables

Acknowledgments

We wish to thank DARPA for supporting this project, and particular Dr. Douglas Maughan who was Program Manager of the Fault Tolerant Networks (FTN) program, under which this project was funded. Our thanks also to Dr. Kevin Kwiat, our Air Force Point of Contact, for making this a fun project to work on.

Summary

We examined the use of overlay networking in conjunction with distributed access control as a mechanism against denial of service (DoS) attacks. In particular, we examine the application of these technologies in allowing undisrupted communication between two peers in the Internet in the face of large distributed denial of service (DDoS) attacks. By design, we avoided changes to protocols and infrastructure (such as routers), and required only minimal cooperation from Internet Service Providers (ISPs). Our goal was to allow the deployment of a solution to the DDoS system that did not depend on large scale adoption by ISPs, and did not require placement of equipment inside the Internet core.

We initially examined the case of well known peers that share some authentication material. In that scenario, the portion of the network immediately surrounding attack targets is protected by high-performance routers that aggressively filter and block all incoming connections from hosts that are not approved, as shown in Figure 1. These routers are “deep” enough in the network (typically in an ISP's Point of Presence), that the attack traffic does not adversely impact innocuous traffic. The identities of the small set of nodes that are approved at any particular time is kept secret so that attackers cannot try to impersonate them to pass through the filter. These nodes are picked from a set of nodes that are distributed throughout the wide area network. This superset forms a *secure overlay*: any transmissions that wish to traverse the overlay must first be validated at any of the entry points of the overlay. This was initially done by requiring users to authenticate to the overlay, as they would to a firewall. In subsequent work, we showed how to relax this restriction through the use of Graphic Turing Tests, that differentiate between humans and automated processes (e.g., DDoS zombies). Once inside the overlay, the traffic is tunneled securely to one of the approved (and secret from attackers) locations that can then forward the validated traffic through the filtering routers to the target.

Thus, there are two main principles behind our design. The first principle is the 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. The second is the ability to recover from random or induced failures within the forwarding infrastructure or the secure overlay nodes.

We demonstrated the efficacy of our approach through a series of simulations and experiments on the real Internet, using the PlanetLab testbed to measure the performance impact of our system on regular communications. We find that our approach can increase the end-to-end latency of a communication flow by a factor of 3, while shielding it from the effects of the DDoS attack. We are continuing research on SOS, in particular seeking to strengthen it against more sophisticated attackers.

Introduction

The Internet is increasingly being used for different kinds of services and interactions with, and between humans. Such an environment provides a rich set of targets for motivated attackers. This has been demonstrated by the large number of vulnerabilities and exploits against web servers, browsers, and applications. Traditional security considerations revolve around protecting the network connection's confidentiality and integrity, protecting the server from break-in, and protecting the client's private information from unintended disclosure. To that end, several protocols and mechanisms have been developed, addressing these issues individually. However, one area that has long been neglected is that of service availability in the presence of denial of service (DoS) attacks, and their distributed variants (DDoS).

In the SOS architecture we address the problem of securing communication in today's existing IP infrastructure from 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 initially 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, 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: (1) 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 (2) 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 arts of the backbone.

Our architecture leverages heavily off of previous work on IP security, IP router filtering capabilities, and novel approaches to routing in overlays and peer-to-peer (P2P) networks. 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 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 to 3, 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.

Methods, Assumptions, and Procedures

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. 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 work.

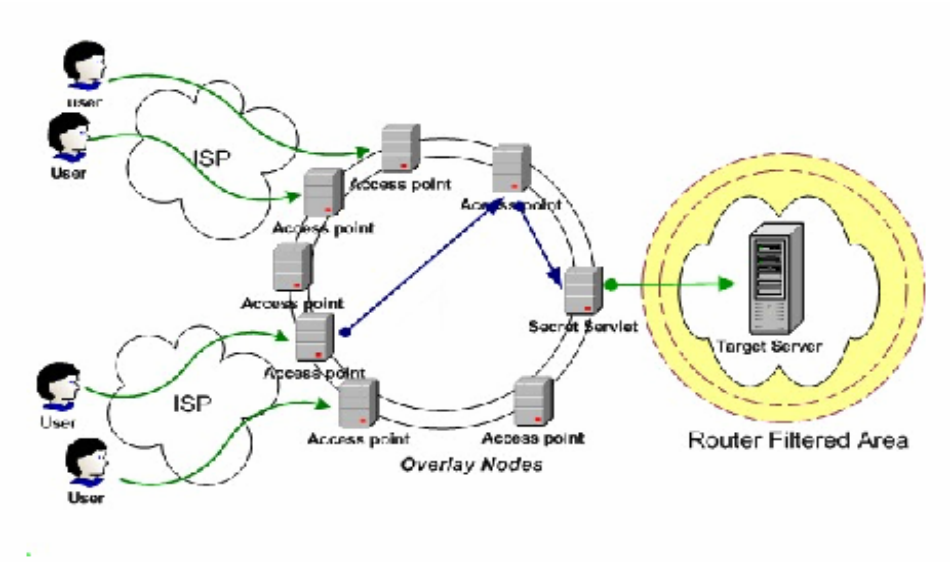

Figure 1: SOS Architecture

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.

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. 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.

Architecture Overview

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

1. A source point that is the origin of the traffic forwards a packet to a special overlay node called a SOAP (Secure Overlay Access Point) that receives and verifies that the source point has a legitimate communication for the target.

2. The SOAP routes the packet to a special node in the SOS architecture that is easily reached, called the beacon.

3. 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.

4. The secret servlet forwards the packet to the target.

5. 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.

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 (1) these routers can handle high loads of traffic, making them difficult to attack, and (2) 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.

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 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. Since that node 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, 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 he 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.

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. Eventually, the packet will arrive at a secret servlet that can then deliver it to the target.

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 has 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 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 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, a list of all SOS nodes may be publicized and used by all clients. We plan to investigate SOAP selection in future work.

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 $O(n/N)$ where N is the number of nodes in the overlay and n 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, 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 is used to map an arbitrary identifier to a unique destination node that is an active member of the overlay.

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 $x + 2^{i-1} \bmod 2^m$. 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. As an example, for *m*=5 and when 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. 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 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 may be sufficient.

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 a secure 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.

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.

Results and Discussion

In order to quantify the overhead associated with use of SOS, we created a simple topology running on the local network (100 Mbit fully-switched Ethernet). For our local-area network overlay, we used 10 commodity PCs running Linux Redhat 7.3. We 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 (at 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).

The browser was located on a separate ISP The reason for this configuration was to introduce some latency in the first-hop connection (from the browser to the SOAP), thus simulating (albeit using a real network) an environment where the browsers have slower access links to the SOAPs, relative to the links connecting the overlay nodes themselves (which may be co-located with core routers). By placing all the overlay nodes in the same location, we effectively measure the aggregate overhead of the SOS nodes in the optimal (from a performance point of view) case.

Server	Direct	1 node	4 nodes	7 nodes	10 nodes
Yahoo!	1.39	2.06	2.37	2.79	3.33
Verisign	3.43	4.22	5.95	6.41	9.01
Columbia BB	0.64	0.86	1.06	1.16	1.21
Columbia BB (2nd)	0.14	0.17	0.19	0.20	0.25

Table 1: SOS on the local network

Table 1 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 BB (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.

As the figure shows, SOS increases the end-to-end latency between the browser and the server by a factor of 2 to 3. These results are consistent with our simulations of using SOS in an ISP topology, where the latency between the different overlay nodes would be small, as discussed in Section 3. The increase in latency can be primarily attributed to the network-stack processing overhead and proxy processing at each hop. It may be possible to use TCP splicing or similar techniques to reduce connection handling overhead, since SOS performs routing on a per-request basis. Furthermore, there is an SSL-processing overhead for the interoverlay communications. A minor additional cryptographic overhead, relative to the direct access case, is the certificate validation that the SOAPs have to perform, to determine the client's authority to use the overlay, and the SSL connection between the proxy running on the user's machine and the SOAP. Such overheads are typically dominated by the end-to-end communication overheads. Use of cryptographic accelerators can further improve performance in that area. One further

optimization is to maintain persistent SSL connections between the overlay nodes. However, this will make the task of the communication module harder, as it will have to parse HTTP requests and responses arriving over the same connection in order to make routing decisions.

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 BB (2nd)	0.14	0.23	0.28	0.57	0.72

Table 2: SOS on PlanetLab

Table 2 shows the same experiment using PlanetLab, a wide-area overlay network testbed. The PlanetLab nodes are distributed in academic institutions across the country, and are connected over the Internet. We deployed our SOS proxies on PlanetLab and ran the exact same tests. Naturally, the direct-contact case remains the same. 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. In each case, the increase in latency over the local-Ethernet configuration 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 (or even 10, in the worst case) is considerable, we believe it is more than acceptable in certain environments and in the presence of a determined attack.

Table 3 shows the results when a shortcut implementation (whereby the full Chord overlay is only traversed when determining the location of the secret servlet, with the actual traffic being directly routed from the SOAP to the secret servlet) was tested on the PlanetLab testbed. This variant provides significant performance improvements, particularly on subsequent requests for the same site, because of the caching. To simulate the effects of an attack on individual nodes in

the overlay, we simply brought down specific nodes. The system healed itself within 10 seconds.

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 BB (2nd)	0.14	0.55	0.56

Table 3: SOS with shortcut routing

Conclusions

The Secure Overlay Services (SOS) architecture allows legitimate users to access a remote site in the presence of a denial of service attack. The architecture uses a combination of Graphic Turing Tests, cryptographic protocols for data origin and principal authentication, packet filtering, overlay networks, and consistent hashing to provide service to both known (a prior) and casual (e.g., web-browsing) users. Our analysis and experiments demonstrate the effectiveness of the architecture to provide service when under attack, while imposing a modest performance penalty to communication flows that traverse it.

More information on the architecture, experiments, and prototype implementation may be found (along with a list of publications) at http://nsl.cs.columbia.edu/projects/sos/

Recommendations

We believe that we have demonstrated that overlay-based mechanisms are an effective mechanism for providing service in the presence of distributed denial of service attacks. Their advantages are the ease of deployment (since they require no changes to infrastructure or protocols, and may be deployed without the cooperation of ISPs) and small impact on performance.

We recommend that DARPA promote further research in this area, and pursue a deployment of a SOS-like overlay network in the DoD environment to gain further insights on the use of the system with realistic workloads, and as a safeguard against DoS attacks.

List of Symbols, Abbreviations, and Acronyms

SOS:	Secure Overlay Services
DoS:	Denial of Service
DDoS:	Distributed Denial of Service
ISP:	Internet Service Provider
SOAP:	Secure Overlay Access Point
P2P:	Peer-to-Peer (network)
IP:	Internet Protocol
POP:	Point of Presence
TLS:	Transport Layer Security
SSL:	Secure Socket Layer
BGP:	Border Gateway Protocol
IPsec:	IP Security
TCP:	Transmission Control Protocol