Using Overlays to Improve Network Security

Angelos D. Keromytis, Vishal Misra, Daniel Rubenstein

Columbia University in the City of New York
{*angelos,misra,danr*}@cs.columbia.edu

ABSTRACT

As we increase our dependency upon networked communication, the incentive to compromise and degrade network performance increases for those who wish to disrupt the flow of information. Attacks that lead to such compromise and degradation can come in a variety of forms, including distributed denial of service (DDoS) attacks, cutting wires, jamming transmissions, and monitoring/eavesdropping. Users can protect themselves from monitoring by applying cryptographic techniques, and the recent work has explored developing networks that react to DDoS attacks by locating the source(s) of the attack. However, there has been little work that addresses preventing the other kinds of attacks as opposed to reacting to them. Here, we discuss how network overlays can be used to complicate the job of an attacker that wishes to prevent communication. To amplify our point, we focus briefly on a study of preventing DDoS attacks by using overlays.

1. INTRODUCTION

Ongoing advances in communication networks continue to dramatically change our ability to "keep in touch". Cell phones, pagers, and wireless handheld devices will soon allow people to communicate regardless of their respective locations, and via several different means including voice, video, and text. The World Wide Web and recent peer-to-peer technologies allow Internet users to locate vast quantities of information and media from their computer. While new technologies are often first viewed as a convenience that simplifies life, people quickly grow dependent on them and develop a reliance on the availability of these services. In addition, the Internet is being used for increasingly time-critical applications (*e.g.*, electricity production monitoring and coordination between different generators), such that there is a growing need for the network to be able to offer high, predictable levels of service.

This reliance increases the incentives of those who wish to disrupt peoples' lives to compromise, damage, and degrade the networks that provide these services. While the Internet was designed to handle naturally occurring failures such as router outages or accidental cuts in wiring, it was not designed to handle intentional attacks like DDoS attacks: attacks that attempt to overwhelm the processing or link capacity of the target site (or routers that are topologically close) by saturating it with bogus packets.

Presumably, the initial designers of the Internet did not foresee the magnitude of society's dependence on their invention, and hence could never imagine there would be those who would actively seek to attack the infrastructure. Since it was assumed that the Internet would operate in "friendly" environments, the fundamental design principles that guided its design assumed all parts of the network would offer service in an attempt to improve performance. This allowed for a looser, less secure architecture where complex decisions were made at the edges of the network and the core of the network performed comparatively simple tasks, essentially trusting the edges to do the "right thing". This principle, commonly referred to as the "end-to-end principle"[1, 2] has been the basic premise behind protocol design. It is often cited as the philosophy that allowed the Internet to survive to support such a wide variety of network applications.

Clearly, there remain numerous benefits to designing a network that follows this philosophy:

- It is easier to replace and update forwarding components (routers) inside the network since there are fewer requirements to satisfy.

This research was supported in part by NSF grants ANI-0117738 and CAREER Award No. ANI-0133829 and by DARPA under contract number F30602-02-2-0125. 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.

- Limiting code to simple tasks reduces the likelihood of software/hardware bugs or algorithmic flaws that bring down the network.

- Extending the network is easier since there are fewer requirements that must be met before an extension is permitted.

However, as has been demonstrated in the past few years,[3–5] the existing end-to-end mechanisms are inadequate at protecting the network from attacks the network.

The need to protect against or mitigate the effects of DoS attacks has been recognized by both the commercial and the research world, and some work has been done towards achieving these goals.[6–9] These mechanisms focus on detecting the source of DoS attacks in progress and then countering them, typically by "pushing" some filtering rules on routers as far away from the target of the attack (and close to the source) as possible. Thus far, the focus of DoS-countering mechanisms has been on *reaction*. The motivation behind this approach has been twofold: first, it is conceptually simple to introduce a protocol that will be used by a relatively small subset of the nodes on the Internet (*i.e.,* ISP routers), as opposed to requiring the introduction of new protocols that must be deployed to and used by end systems. Second, these mechanisms are fairly transparent to protocols, applications, and legitimate users.

Hence, the networking community has reached a dilemma. Do we abandon the philosophy of keeping complexity out of the core of the network? Do we require security measures and standards at every router everywhere to protect against on-line sabotage? Our position in this paper is that often it is not necessary to make such modifications that might complicate future deployment and modification. Instead, we suggest that researchers consider using network overlays as an alternative. A network overlay is a virtual network constructed upon an underlying, physical network. The participating end-systems act as routers within this virtual network. A *link* between two end-systems in the overlay network is formed from a physical path (*e.g.,* the IP route) that connects these two end-systems. Hence, wherever there is a physical path in the underlying network, there can exist a link in the overlay network. Having so many available links increases the flexibility of the network, and a more flexible network is less likely to be susceptible to attacks.

Of course, one should not overlook the fact that this overlay network still resides on a physical network. It stands to reason that if an attacker can bring down the physical network, then the overlay will be brought down as well. This observation led us to consider the following question:

- Can an overlay provide additional security beyond that which can be guaranteed by the underlying physical network, or is the overlay as weak as the weakest point of the underlying network?

In what follows, we explore these questions by first considering some of the recent developments in overlay routing, and claim that indeed, the approaches used upon the overlay cannot be achieved directly upon the underlying network. We then turn our attention specifically on security. We argue that overlays provide stronger security by increasing the potential set of paths a flow can take through the underlying network, which in turn complicates the procedure that would be undertaken by an attacker that wishes to sabotage the communication. As a particular case study, we point to our work on *Secure Overlay Services (SOS)*[10]it, where we use overlays to allow a site to receive traffic from pre-approved, authorized users, while reducing the likelihood of an attacker launching a successful DDoS attack to a negligible probability.

The paper proceeds as follows. In Section 2, we discuss related work on overlay approaches and explore why these approaches could not be performed on the underlying network. In Section 3, we explore several security problems, and consider how overlay networking might be used to address these problems. In Section 4, we briefly explore our work on SOS which demonstrates an overlay-based security solution. In Section 5 we list some open problems involving security and overlays, and we conclude in Section 6.

2. USING NETWORK OVERLAYS

In this section, we attempt to ascertain the novel features of an overlay network. In particular, we wish to examine what are the essential properties that make an overlay network different from a physical network.

One of the initial uses of overlay networking was to enable multicast within the network after several years of emphasis on building a wide-area, network-layer (router-centric) multicast service.[11–14] While network layer approaches showed promise within the research community, the economic structure of the Internet prevented their wide-scale deployment and acceptance.[15] As a result, researchers turned their focus to an application-layer multicast.[16–18] There, multicast trees were formed atop overlays: several end-systems would act as multicast forwarding points, sending copies of the transmission simultaneously to several additional end-systems. Furthermore, these trees could be adjusted dynamically to meet the needs of applications much more easily than in their network-layer multicast alternative.[19,20]

Overlays also found use in improving the performance and robustness of unicast routing[21,22] by providing alternate paths from a particular source to a particular destination along paths that proceeded through intermediate end-systems. It was shown that, despite the additional overhead of passing through multiple end-system nodes, it was often the case that such alternate paths would provide routes with smaller source-to-destination delivery latencies than the underlying, direct IP path. In addition, an alternate path for communication would often remain active when routing anomalies caused disruptions along the underlying IP path.

Most recently, overlays have seen wide deployment as a means of exchanging content between peers. Peer-to-peer (P2P) networks such as Gnutella[23] are overlays where peers - the end-systems that participate in the protocol - assist one another by forwarding or responding to search requests for content. The initial design of protocols that performed these searches essentially broadcast requests upon the overlay. More recent work has produced approaches that reduce the overhead of searches that, for each content object, provides a means for identifying a peer in the overlay that is responsible for storing or knowing the storage location of that object, and also provides a distributed means of routing to that peer with that knowledge.[24–27]

All of these works make use of two fundamental properties of an overlay network that do not exist in the underlying physical network.

- **Full connectivity of the overlay graph**:In an overlay network, any (virtual) router can connect to any other (virtual) router. In contrast, in a physical network, topological position limits the sets of pairs of routers that can be directly connected. Hence, physical limitations bound the number of neighbors of a router and, in addition, a router's set of neighbors can be considered static. In an overlay, what limits the set of neighbors is not physical, but rather is the communication complexity involved in maintaining connections between all pairs of participants in the overlay. Overlay protocols therefore focus on choosing the "right" set of neighbors.
- **Dynamics in the membership of virtual routers**: Once a router is installed inside the network, it is expected to perform its routing function whenever called upon to do so, and only ceases to perform this task when a) it is explicitly disconnected from the network or b) there is some unanticipated failure, In contrast, protocols run atop overlays expect frequent changes in topology, as end-systems enter and exit from participating within a specific application. Gnutella is a perfect example of this phenomenon. When users run Gnutella, their machine joins the overlay and acts as a virtual router. When the user terminates the Gnutella application, the machine leaves the overlay and ceases to behave as a virtual router.

We now turn our attention to our first question. Can overlays provide additional security beyond what is provided by the underlying physical network? The previous work discussed above indicates that overlays can improve network performance (reducing delay and recovering from component failures) and provide classes of applications that are not necessarily provided by the underlying network (multicast capabilities, content search and discovery). Clearly, overlays can perform tasks beyond that of the network upon which they exist. So how can the properties above be used to help secure the network?

- **Robustness**: Since overlays increase a network's robustness, attackers will have a more difficult time bringing down the network.
- **Dynamics**: Since participants of the overlay come and go, there is a greater challenge in even deciding what to attack.
- **Increased Alternatives**: Network routing typically provides a single path between two points. In contrast, overlay routing offers a seemingly limitless set of paths between two points. Thus, an attacker that wishes to bring down a particular communication must be prepared to attack a much larger portion of the network when overlays are used.

3. SECURITY PARADIGMS

Traditionally, the security problems inherent in an open network such as the Internet have been:

- **Data Confidentiality:** keeping communication contents secret from potential eavesdroppers.
- **Data Integrity:** ensuring that what was received by the recipient of a message is what was originally sent by the sender.
- **Privacy:** "scrubbing" different sessions by the same user such that they cannot be correlated (for traffic analysis, marketing, or spying purposes).
- **Authentication:** determining or verifying the identity of an entity (user, network node, *etc.*).
- **Access Control:** restricting access to sensitive data or resources to only authorized users.

Notice that some of these considerations are inter-dependent: maintaining a user's privacy involves (but is not limited to) keeping the contents of his communications confidential; user authentication is typically followed by use of protocols or mechanisms that ensure the integrity of the communication (to prevent session hijacking, or similar attacks). Authentication itself is usually a prerequisite to access control.

At first, it may seem that overlay networks are not particularly applicable to these problems, especially since security is considered an end-to-end concern (similar to networking). However, overlay networks have been proposed, and some have been in use, for many years in addressing some of these problems.

The area which has attracted the most attention, and has arguably seen the most effective application of overlay networking has been that of privacy. The original idea of email *mixes*[28] is perhaps the earliest use of overlays for providing anonymous (and pseudonymous) email communication; it made use of a network of email servers that successively decrypt email messages and forward them to the next node (another mix node, or the final destination). Refinements of this technique have been proposed and are currently in use (*e.g.,* the mix-master anonymizing network).

Subsequently, the idea of mixes was proposed for anonymizing calls over an ISDN network,[29] and later on adopted for web browsing anonymity in the Crowds system.[30] Onion Routing[31] generalized the concept, to provide anonymous connections between any pair of hosts and for almost any application. While the mechanisms employed by these systems differ, reflecting different threat models and engineering trade-offs during their design, they all make use of a set of nodes that are not topologically adjacent, and which handle traffic on behalf of a number of communicating parties. In some cases (*e.g.,* Crowds), the system includes the users' systems; this was an early example of peer-to-peer overlay networking. Generally, overlay networks can be used to hide traffic patterns; as we mentioned in Section 2, the work factor of the attacker increases with the number of nodes that participate in the overlay.

The areas of confidentiality, authentication, and access control have also seen the use of some form of network overlays: the so-called "intranets" and "extranet", typically based on network-level security protocols such as IPsec,[32] bring together geographically (and topologically) disjoint networks and present the illusion of a single network. Here, the role of the specialized overlay nodes is played by firewalls and remote-users' workstations. Similar techniques can be used to ensure that sensitive traffic does not traverse untrustworthy link or routers. Admittedly, the usefulness of overlays seems limited in this area — primarily because confidentiality and integrity seem to, fundamentally, be end-to-end considerations.

More recently, denial of service (DoS) attacks, and their distributed variants, have also become a major concern. Such attacks seek to overcome the capacity of the target's link by saturating it with bogus (but usually valid-looking) traffic. In the next section, we describe the *Secure Overlay Services (SOS)* architecture, designed to mitigate the effects of such attacks.

4. SOS: A SECURE OVERLAY SERVICE SOLUTION

We review the SOS architecture proposed in.[10] The goal of the SOS architecture is to allow communication between a *confirmed source point* and a *target*. By confirmed, we mean that the target has given prior permission to a specific user that currently resides at the source point. Typically, this means that the source must be authenticated and authorized by the SOS infrastructure before traffic is allowed to flow between itself and the target through the overlay.

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

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

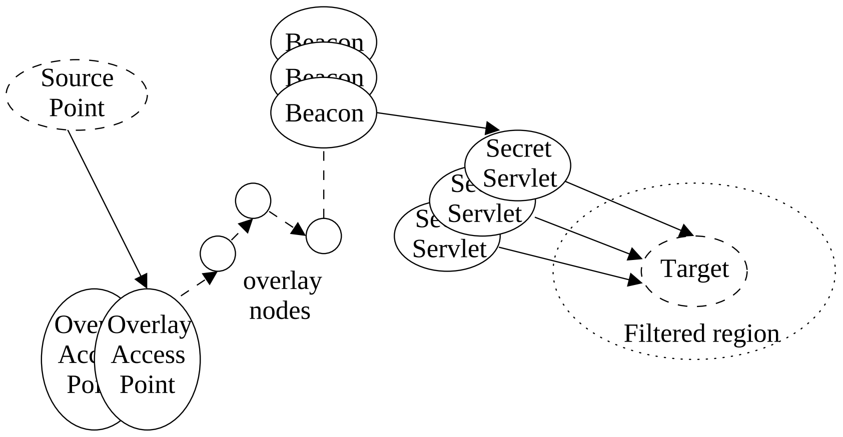

Figure 1. Basic SOS architecture.

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

- **Targets**: Target nodes wish to receive transmissions from validated sources and wish to be protected from phony (*i.e.,* un-authenticated) transmissions. Heavy filtering is applied in the immediate vicinity of the target to protect it from unwanted traffic. We assume that attackers do not have access to routers inside the filtered region (*i.e.,* they cannot observe which source addresses can proceed through the filter). Past history indicates that it is a lot more difficult for an attacker to completely take over a router or link in the middle of an ISP's network than to attack an end-host; intuitively, this is what we would expect, given the limited set of services offered by a router (compared to, *e.g.,* a web server or a desktop computer).
- **Secret Servlets**: Nodes that participate on the overlay and act as the (only) entry point to a target. Their identities are kept as secret as possible.
- **Beacons**: A beacon is a node that participates on the overlay. It receives traffic destined for a particular target and, after verifying the legitimacy of the traffic, forwards it to a secret servlet. Hence, beacons are aware of the identities of some of the secret servlets for the targets for which they act as a beacon.
- **Overlay Access Point (OAP)**: A node that participates on the overlay that accepts traffic from "approved" source points that wish to use the overlay to reach a given destination.

- **Source points**: A node on or off the overlay that wishes to send a (legitimate) transmission to a target. It is assumed that source points have been granted permission by the target during an earlier exchange (*e.g.*, have received an appropriate certificate through e-mail).
- **Attack point**: Any node that has been compromised and can be used to launch an attack or snoop the source from where a packet came or destination to where a packet is going (both next hop and final).

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

4.1. Filtering at the target

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

When using network overlays, it is still possible for a packet of arbitrary origin to reach the target even when intensive filtering is applied (without spoofing). This is accomplished by forwarding traffic to locations in the overlay whose addresses are permitted to pass through the filter. These locations then forward traffic through the filter. To provide security, two properties must hold:

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

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

4.2. Secret Servlets

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

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

4.3. Access Points

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

A legitimate client chooses a node N_A from a list of access points and initiates a secure communication with that node using a protocol such as IPsec.[32] Hence, when N_A agrees to act as the access point for this client, it has confirmed both the client's right to communicate with the target as well as the IP address of the client. Subsequent traffic between the access point and the client may be protected (again, by IPsec). As an alternative, the client may be given a "cookie" that it has to include into any subsequent packets it sends to the access point. The choice between the two mechanisms depends largely on the perceived threat model: if attackers cannot eavesdrop the communication path between access points and sources, then the cookie approach is sufficient. In a more hostile environment (*e.g.,* when using a wireless network to connect to the Internet), a cryptography-based mechanism may be used.

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

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

4.4. Overlay Routing

Routing used within SOS must be robust to attacks upon nodes within the overlay. In particular, if a given set of overlay nodes is attacked, there must be an efficient transition to an alternative path where no nodes are under attack. One approach is to apply the consistent hashing approach used within the Chord service.[24] For our purposes, Chord can be viewed as a routing service with the following properties:

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

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

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

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

4.5. Summary of Architecture

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

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

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

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

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

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

This scheme is robust against DoS attacks because:

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

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

- If a secret servlet is attacked (either due to a lucky random hit or somehow its identity was compromised), the secret servlet (which can still send traffic outward) can notify the target and the target can choose alternate secret servlets.

- If a secret servlet's identity is discovered and attacks arrive at the target with the source IP address of some secret servlet, the target can choose an alternate set of secret servlets.

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

5. DISCUSSION

We have argued that overlays can be used to address security flaws of the Internet. However, security is an adversarial game, in which every time a new solution is provided, that solution often has holes or oversights that an attacker can exploit. We address this concern in this section. We consider several challenges that are hurdles toward applying overlays to secure networks.

- **Attacks from inside the overlay**: There are certain pieces of information that must be shared by overlay participants that we have assumed can be kept hidden from attackers. For instance, these include private keys that nodes use to validate their identity, or path information about an active communication that is to be secured. Were an attacker to gain access to this information, there would be little benefit to the randomness and unpredictability that the overlay provides. Thus, it is essential that attackers be kept from taking part in the overlay process. In this regard, it is important to set up some form of "neighborhood watch" where overlay nodes monitor one another for erratic behavior. When an overlay node's behavior deviates from the norm, that node should be evicted from the session. Hence, there needs to be some form of consensus-building protocol and, if the consensus is that a participant should be removed from the overlay, then the mechanism should exist to evict that participant based on consensus.
- **A shared secure overlay**: The security offered by an overlay increases with the number of participants that form the overlay. This is because more participants provide more options for potential path selection. However, for some applications, there is a tendency to build separate, small overlays that operate simultaneously, using the same set of network resources, but not interacting at the application level. Multicast overlays are one such example, where each instance of an application that utilizes multicast would build its own overlay tree. In the interest of security, this separation facilitates successfully attacking a specific instance that an attacker might be interested in. Thus, it may prove useful to have a means for several instances of the same application (or perhaps even across different applications) to somehow merge their overlays together to provide additional security.
- **Timely delivery**: Since the security provided by using overlays depends upon the magnitude of the variety of paths, it is important that applications wishing high levels of security accept transmission over a large set of paths. Since the number of paths that offer optimal performance (*e.g.,* low delay) can be small, applications may have to sacrifice timely delivery in order to obtain a reasonable level of security. Determining the right levels of security and delay is an interesting open problem.

6. CONCLUSION

We have explored how overlays can be applied to improve security within network systems. By increasing the robustness, flexibility, and unpredictability of a system, the system becomes harder to attack. To demonstrate our point, we described our SOS architecture, which is designed to provide DDoS-proof communication from a set of legitimate users to a target. We point out a set of open problems in securing networks through the use of overlays, and hope that our discussion here will promote additional interest in the area.

REFERENCES

1. J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-to-end arguments in System Design," *ACM Transactions on Computer Systems* **2**, pp. 277–288, November 1984.
2. J. Saltzer, D. Reed, and D. Clark, "End-to-end arguments in system design," *ACM Trans. on Computer Systems* **2**, pp. 277–288, November 1984.
3. S. E. I. C. E. R. Team, "CERT Advisory CA-96.21: TCP SYN Flooding and IP Spoofing Attacks," tech. rep., SEI, Sept 1996. ftp://info.cert.org/pub/cert_advisories/CA-96.21.tcp_syn_flooding.
4. C. Schuba, I. Krsul, M. Kuhn, E. Spafford, A. Sundaram, and D. Zamboni, "Analysis of a denial of service attack on tcp," in *IEEE Security and Privacy Conference*, pp. 208–223, May 1997.
5. L. Heberlein and M. Bishop, "Attack Class: Address Spoofing," in *Proceedings of the 19th National Information Systems Security Conference*, pp. 371–377, October 1996.
6. J. Ioannidis and S. M. Bellovin, "Implementing Pushback: Router-Based Defense Against DDoS Attacks," in *Proc. of the Network and Distributed System Security Symposium (NDSS),* February 2002.

7. D. Dean, M. Franklin, and A. Stubblefield, “An Algebraic Approach to IP Traceback,” in *Proc. of the Network and Dsitributed System Security Symposium (NDSS)*, pp. 3–12, February 2001.
8. S. Savage, D. Wetherall, A. Karlin, and T. Anderson, “Network Support for IP Traceback,” *ACM/IEEE Transactions on Networking* **9**, pp. 226–237, June 2001.
9. S. Savage, D. Wetherall, A. Karlin, and T. Anderson, “Practical Network Support for IP Traceback,” in *Proc. of the 2000 ACM SIGCOMM Conference*, pp. 295–306, August 2000.
10. A. Keromytis, V. Misra, and D. Rubenstein, “SOS: Secure Overlay Services,” in *Proceedings of ACM SIGCOMM’02*, (Pittsburgh, PA), August 2002.
11. S. Deering and D. Cheriton, “Multicasting routing in datagram internetworks and extended LANs,” *ACM Transactions on Computer Systems* **8**, pp. 85–110, May 1990.
12. T. Ballardie, P. Francis, and J. Crowcroft, “Core based trees (CBT),” in *Proceedings of ACM SIGCOMM’93*, pp. 85–95, (San Francisco, CA), September 1993.
13. S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C. Liu, and L. Wei, “An Architecture for Wide-Area Multicasting,” in *Proceedings of ACM SIGCOMM’94*, (London, UK), August 1994.
14. H. Holbrook and D. Cheriton, “IP Multicast Channels: EXPRESS Support for Large-Scale Single-Source Applications,” in *Proceedings of SIGCOMM’99*, (Cambridge, MA), September 1999.
15. C. Diot, B. Levine, B. Lyles, H. Kassem, and D. Balensiefen, “Deployment Issues for the IP Multicast Service and Architecture,” *IEEE Network Magazine* , Jan/Feb 2000.
16. Y. Chu, S. Rao, and H. Zhang, “A Case for End System Multicast,” in *Proceedings of ACM SIGMETRICS’00*, (Santa Clara, CA), May 2000.
17. J. Jannotti, D. Gifford, K. Johnson, M. Kaashoek, and J. O’Toole, “Overcast: Reliable Multicasting with an Overlay Network,” in *Proceedings of USENIX*, (San Diego, CA), October 2000.
18. Y. Chawathe, S. McCanne, and E. Brewer, “RMX: Reliable Multicast for Heterogeneous Networks,” in *Proceedings of IEEE INFOCOM’00*, (Tel-Aviv, Israel), March 2000.
19. Y. Chu, S. Rao, S. Seshan, and H. Zhang, “Enabling Conferencing Applications on the Internet Using an Overlay Multicast Architecture,” in *Proceedings of ACM SIGCOMM’01*, (San Diego, CA), August 2001.
20. Z. Liu, N. Malouch, D. Rubenstein, and S. Sahu, “Delay-bounded End-system Multicast with Proxy Support,” in *Proceedings of the Tenth International Workshop on Quality of Service (IWQoS)*, May 2002.
21. S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson, “The End-to-End Effects of Internet Path Selection,” in *Proceedings of SIGCOMM’99*, (Cambridge, MA), September 1999.
22. D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, “Resilient Overlay Networks,” in *Proceedings of ACM SOSP’01*, (Banff, Canada), October 2001.
23. “The Gnutella Protocol Specification v0.4, revision 1.2.” Available from http://gnutella.wego.com.
24. I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, “Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications,” in *Proceedings of ACM SIGCOMM’01*, (San Diego, CA), August 2001.
25. F. Dabek, , F. Kaashoek, R. Morris, D. Karger, and I. Stoica, “Wide-Area Cooperative Storage with CFS,” in *Proceedings of ACM SOSP’01*, (Banff, Canada), October 2001.
26. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, “A Scalable Content-Addressable Network,” in *Proceedings of ACM SIGCOMM’01*, (San Diego, CA), August 2001.
27. A. Rowstron and P. Druschel, “Storage Management and Caching in PAST, A Large-scale, Persistent Peer-to-peer Storage Utility,” in *Proceedings of ACM SOSP’01*, (Banff, Canada), October 2001.
28. D. Chaum, “Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms,” *Communications of the ACM* **24**(2), pp. 84–90, 1981.
29. A. Pfitzmann, B. Pfitzmann, and M. Waidner, “Isdnmixes: Untraceable communication with very small bandwidth overhead,” 1991.
30. M. K. Reiter and A. D. Rubin, “Crowds: anonymity for Web transactions,” *ACM Transactions on Information and System Security* **1**(1), pp. 66–92, 1998.
31. M. G. Reed, P. F. Syverson, and D. M. Goldschlag, “Anonymous connections and onion routing,” *IEEE Journal on Special Areas in Communications* **16**(4), pp. 482–494, 1998.
32. S. Kent and R. Atkinson, “Security Architecture for the Internet Protocol,” Request for Comments (Proposed Standard) 2401, Internet Engineering Task Force, Nov. 1998.