

SOS: Secure Overlay Services

Angelos D. Keromytis, Vishal Misra, Dan Rubenstein
Department of Computer Science, Department of Electrical Engineering
Columbia University, New York, NY

Paper report

Yannis Klonatos - S.N. 583

May 29, 2010

Introduction (The proposed scheme)

We want to allow moderate number of ***legitimate users*** (who are authenticated in advance) to communicate with ***a target destination***, where:

- DoS attackers will attempt to stop communication to the target
- The target is difficult to replicate (e.g., info highly dynamic),
- Legitimate users may be mobile (source IP address may change)

Such a scenario is *not* uncommon or surreal since there are cases in real life that require it. For instance, FBI/Police/Fire personnel in the field may need communicating with their agency's database. Another example is of bank users' that commonly access their banking records and on-line customers completing a transaction in a e-shop.

SOS: Basic Idea

DoS attacks are effective because of their many-to-one nature: many machines attack one (DDoS attacks). The basic idea behind SOS is to send traffic across an *overlay*: a virtual network whose "links" are routing paths in the underlying physical network. By doing so we achieve to:

1. Force attackers to need to attack many overlay points in order to mount successful attack. Since the paths to the target are numerous and all hidden, all paths need to be brought down.
2. Allow our network to adapt quickly: the "many" that must be attacked can be changed.

Ideally, we would like a solution that is easy to distribute: and doesn't require modifications in all network routers (this would be an unreasonable requirement). Furthermore, we would like our solution to not require high complexity (e.g., crypto) operations at/near the target.

One Important assumption the authors make is that an attacker cannot succeed in a DoS attack to the core network routers and can only simultaneously attack a bounded number of distributed end-systems.

SOS: Building up the Solution step-by-step

- **Step 1: Source Address Filtering**

A first step to protect the target is to have routers “near” the target apply simple packet filter based on IP address. As a result, legitimate users’ IP addresses are allowed through, but illegitimate users’ IP addresses aren’t. This is a very fast and light-weighted procedure.

However, there are multiple problems with this solution since bad users may have the same IP address with good users. Or bad users may know good user’s IP address and employ IP spoofing. Finally, there is also the case that the good IP addresses changes frequently (mobility), thus requiring frequent filter updates.

- **Step 2: Filtering + Proxies**

To resolve the above problem, the authors place a proxy server between a legitimate user and the target. The proxy only forwards *authenticated* packets and only packets from the proxy can reach the target .

However, there again some problems since if the attacker learns the IP of a proxy, then he can spoof packets with this address and successfully reach the target. Furthermore, attackers can directly DoS attack on the proxy to drag it down, thus, again preventing legitimate user communication.

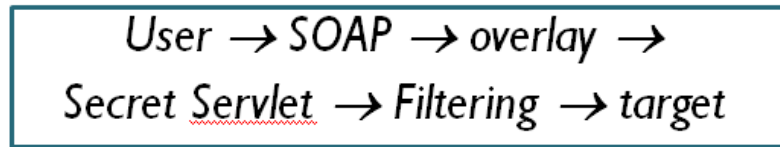
- **Step 3: Filtering + Hidden proxy server**

To resolve the above problems, the authors propose to just keep the identity of the proxy “hidden”. The hidden proxy called a ***Secret Servlet and*** only the target, the secret servlet itself, and a few other authenticated users in the network know the secret servlet’s identity (IP address). Thus, attacker **cannot** be sure which node is a proxy for the target.

- **Step 4: Filtering + Hidden Proxy + Overlay Routing + SOAP**

The authors then claim that additional security can be obtained by sending traffic to the secret servlet via a *network overlay*. Nodes in virtual network are often end-systems and verification/authentication of “legitimacy” of traffic can be performed at each overlay end-system hop (if/when desired). This requires advertising a set of nodes that can be used by the legitimate user to access the overlay. These access nodes

that participate within the overlay are called **Secure Overlay Access Points (SOAPs)**. By adding SOAPs the path of SOS is now as follows:

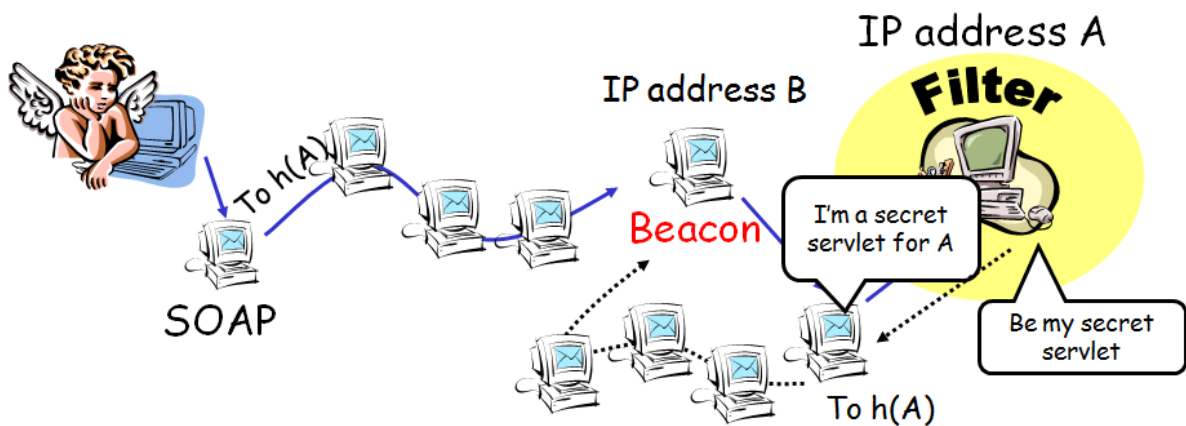


- **Step 5: Filtering + Hidden Proxy + Overlay Routing + SOAP + Chord**

With filters, multiple SOAPs, and hidden secret servlets as described above, we have managed to confuse the attacker: He now *does not have a clue* of what the entry point to the network is. However, we must still get from SOAP to Secret Servlet in a “hard-to-predict manner”. Random routing doesn’t help since the routing complexity is $O(n)$ and routes should not “break” as nodes join and leave the overlay (i.e., nodes may leave if attacked).

For these problems, the authors propose the use of DHT (Dynamic Hash Table) routing that comes with the distributed protocol Chord. SOS (via Chord) now utilizes a Beacon to go from overlay to secret servlet. The Secret Servlet is chosen by target (arbitrarily). Legitimate user forwards packet to SOAP, and SOAP forwards in turn verified packet to Beacon (via Chord). Finally Beacon forwards verified packet to secret servlet, so that finally the Secret Servlet forwards verified packet to target.

The overall approach is shown in the following figure:



Adding Redundancy in SOS

In SOS, each special role can be duplicated if desired: Any overlay node can be a SOAP, the target can select multiple secret servlets and multiple Beacons can be deployed in CHORD. Thus, an attacker that successfully attacks a SOAP, secret servlet or beacon in fact brings down **only** a subset of connections, and **only while** the overlay detects and adapts to the attacks.

Why attacking SOS is difficult

To begin with, if one attacks the target directly, without knowing the secret servlet ID, we have the filter to protect the target. Furthermore, it is impossible for one to attack the secret servlets, since they are hidden. Moreover, attacked servlets automatically “shut down” and new servlets are selected. Attacked beacons also “shut down” (they leave the overlay) and new nodes automatically become beacons. As a result the attacker must continuously attack a “shut down” node or it will return to the overlay at some point in the near future. Finally, if one attacks other overlay nodes, then the nodes themselves can shut down or leave the overlay and the routing in the network self-repairs.

Results

The authors show in their results that for a **Static attack**, where an attacker chooses M of N nodes at random and focuses attack on these nodes, the attacker must bring down almost all overlay nodes to achieve a high likelihood of DoS, and that the more the beacons in the system, the less the likelihood of a successful DoS attack.

Furthermore, the authors investigate the case of a **Dynamic Attack** where an ongoing attack/repair battle takes place: 1) SOS detects & removes attacked nodes from overlay, repairs take time T_R , 2) Attacker shifts from removed node to active node, detection/shift takes time T_A (freed node rejoins overlay). They show that if T_A is significantly smaller than T_R the attacker can take down the SOS infrastructure.

Conclusions

The proposed system, SOS, achieves in protecting a target from DoS attacks by letting ONLY legitimate (authenticated) users through. The approach is based on placing a filter around the target, allow only “hidden” proxies to pass through the filter, and the use of network overlays to allow legitimate users to reach the “hidden” proxies. Analysis results show that an attacker without overlay “insider” knowledge must attack a significant amount of overlay nodes to deny service to target.