A New Perspective in Defending against DDoS

Shigang Chen Randy Chow
Department of Computer & Information Science & Engineering
University of Florida, Gainesville, FL 32611, USA
{sgchen, chow}@cise.ufl.edu

Abstract

Distributed denial of service (DDoS) is a major threat to the availability of Internet services. The anonymity allowed by IP networking, together with the distributed, large scale nature of the Internet, makes DDoS attacks stealthy and difficult to counter. As various attack tools become widely available and require minimum knowledge to operate, automated anti-DDoS systems are increasingly important. This paper studies the problem of providing an anti-DoS service (called AID) for general-purpose TCP-based public servers. We design a random peer-to-peer (RP2P) network that connects the registered client networks with the registered servers. RP2P is easy to manage and its longest path length is just three hops. The AID service ensures that the registered client networks can always access the registered servers even when they are under DoS attacks. It creates the financial incentive for commercial companies to provide the service, and meets the need for enterprises without the expertise to outsource their anti-DoS operations.

1 Introduction

The goal of a DoS (denial of service) attack is to completely tie up the resources of a server so that legitimate users are prevented from accessing the service. To conceal the attacker's identity, forged source addresses must be used in the packets sent from the attacker. Moore, Voelker, and Savage's work demonstrated that DoS attacks was widespread in the Internet. By using a novel traffic-monitoring technique, called "backscatter analysis", they observed 12,805 attacks on over 5000 distinct Internet hosts belonging to more than 2000 distinct organizations during a three-week period [4].

Much work against DoS is on anti-address-spoofing. Ferguson and Senie proposed *ingress filtering* [1], which requires the routers of stub networks to inspect outbound packets and discard those packets whose source addresses do not belong to the stub networks. Park and Lee pioneered with the concept of *route-based distributed packet filtering* [5]. The basic idea is for a router to drop a packet if the packet is received from a link that is not on any routing path from the packet's source to the packet's destination. The paper demonstrated that a partial deployment (on 18% of Internet AS's) can effectively prevent spoofed IP packets from reaching their victims. Wang, Zhang and Shin presented a simple and effective mechanism, called *SYN-dog*, to identify SYN flooding sources [8]. It is a software agent installed at leaf routers connecting to stub networks. The agent detects SYN flooding from the attached networks by monitoring the differences between outbound SYN packets and inbound SYN/ACK packets.

For all above approaches, although the deployment can be carried out incrementally, the effectiveness of preventing DoS comes only after the filters or the software are widely deployed across the Internet. An Internet-wide deployment can be difficult to achieve due to political, financial, and administrative reasons, or different technology preferences. On the contrary, it will give individual organizations more incentive to spend the money and resources if they can immediately get the benefit without dependency on the actions of other organizations.

Keromytis, Misra, and Rubenstein proposed a novel architecture called *Secure Overlay Services* (SOS) [3], which proactively prevents DoS attacks. Access requests will be authenticated and routed via a Chord overlay network [6] to one of the servlets, which then forward the requests to the target site. The robustness against DoS comes from the facts that, only authenticated traffic can enter the overlay network, the locations of the servlets are not predictable, and the target site only accepts packets from the servlets. SoS was designed for emergency services or similar types of communication [3]. A certificate for accessing a protected server must be issued to each authorized client. Hence, it is not suitable for protecting a general-purpose public server (such as Yahoo or Google), because all users (including the undercover attackers) are supposed to be authorized, which makes the authentication itself meaningless.

We study the problem of providing an anti-DoS service

(called AID) for general-purpose TCP-based public servers. The AID service ensures that the registered client networks can always access the registered servers even when they are under DoS attacks. The practical advantages of such a service are apparent. A centrally-managed service eliminates the requirement of an Internet-wide deployment campaign involving different administrative domains. It creates the financial incentive for commercial companies to provide the service, and meets the need for enterprises without the expertise to outsource their anti-DoS operations. It supports incremental deployment on a registration basis. A registered client or server immediately receives the protection. AID achieves the following design goals.

1. The service is open to all clients and does not need authentication. It is able to tolerate attacks launched from unregistered clients or from registered clients. Even when a significant portion of registered clients is compromised, the impact on the throughput of the rest clients remains modest.

2. The service works with the existing applications. The registration does not require the upgrade of software, OS, or protocols at the client or at the server. Beside the edge devices of registered networks, no deployment is required on the router software.

3. The service is scalable, efficient, and light-weighted. The service is used only when a DoS attack occurs; when there is no attack, the registered clients and servers communicate directly via the Internet as usual.

4. The service reacts quickly against an on-going DoS attack by blocking out the attack traffic while minimizing the misblocking of legitimate traffic.

5. The service itself is robust against DoS attacks.

None of the existing anti-DoS solutions meet the above goals.

2 Architecture of A Global Anti-DoS Service

2.1 System Architecture

Not every organization has the expertise to implement and manage a complex anti-DoS defense system. Many enterprises in retail, manufactory, finance, or other industries prefer to outsource certain security operations to an independent service provider as long as these operations do not require the disclosure of private information. We describe a global anti-DoS service, called *AID*, which ensures a registered client network the accessibility to any registered server as long as the client does not participate in the attack.

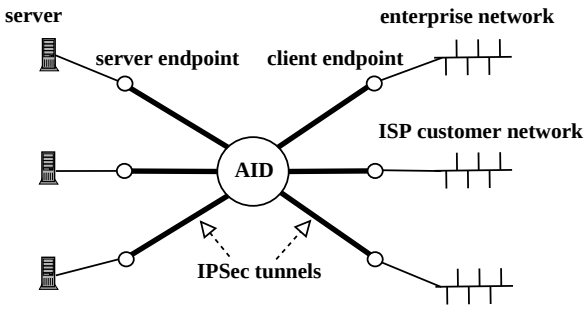

Figure 1. AID system architecture

This protection is voided for those registered clients that host attackers.

The system architecture is shown in Fig. 1. Illustrated as one node, the AID service is implemented as a distributed overlay system consisting of geographically dispersed AID stations for service registration and anti-DoS operations. Unlike many P2P networks where nodes can join from anywhere, we envision the dedicated AID stations are owned, deployed, and managed by a single or a few trusted entities. The number of AID stations can be thousands or tens of thousands. The AID stations communicate among themselves via secure communication channels (such as IPSec tunnels). A client (or server) network may register to a nearby AID station. As part of the registration process, a hardware device or a software module is installed at the edge of the client network to support secure VPN. The registration establishes an IPSec tunnel with the AID station. All tunnels together form an exclusive overlay network between the registered clients and the registered servers. This overlay network will be activated when a registered server is under attack.

An IPSec tunnel has two endpoints, one at the AID station and the other at the edge device of the client (or server) network. A crypto access control list (crypto-ACL) is defined at both tunnel endpoints. It specifies what traffic should be put through the tunnel. Initially the crypto-ACL blocks all user traffic from entering the tunnel. The communication between the clients and the servers are conducted normally via the Internet.

A service process runs at the AID station, and a slave process runs at the edge device of each registered network. The crypto-ACL should permit the control traffic between the two processes. When a server is under attack, it will signal its AID station, which propagates the information via the overlay network to the other AID stations. Each AID station instructs its slave processes to modify the crypto-ACLs to admit a portion (e.g., connection requests only) or all client traffic for the server into the tunnels.

When the AID stations receive the tunneled traffic from the registered clients, they perform distributed scheduling

and route the packets to the server. A distributed virtual-clock scheduling algorithm is used to create the appropriate gaps between the delivery times of consecutive connection requests (or data packets) from the same client network. Each client network receives a fair share of the server's capacity. If a client network does not carry the attack traffic, its requests will be timely processed.

During an attack, part or all communication traffic from the registered clients to the registered servers is carried by the IPSec tunnels. The traffic from the servers to the clients are delivered outside of the tunnels. For servers not under attack, traffic in both directions are through the Internet. Because there is no privacy requirement, IPSec AH is used, which only requires a hash function (MD5 or SHA1) to generate/verify MAC. These hash functions are very efficient and there are software/hardware products operating at the line speed of Gigabit per second, which prevents the tunnel from being a bottleneck.

To prevent the AID stations or the edge devices of registered networks from being the target of DoS attacks, we require the tunneled traffic to be given higher priority. Service contracts should be established with ISP edge routers to forward tunneled traffic ahead of Internet traffic, which shields the network connections of AID stations and edge devices from brute-force DoS attacks launched from the Internet. If the attackers forge tunneled packets, the IPSec endpoints can efficiently drop them due to unexpected sequence numbers in the AH headers.

2.2 Random Overlay Network

Each AID station can take a limited number of VPN tunnels. For example, the high-end Cisco PIX 535 Firewall supports up to 2000 VPN connections [7]. Much of this capacity should be reserved for the registration of client/server networks. The number of tunnels used for connecting AID stations should be minimized. A mesh configuration is clearly not a scalable option because of excess resource consumption, given that the number of AID stations (IPSec-capable devices) may grow into thousands. We propose to form a random overlay network (RON) among the AID stations, which ensures full connectivity with much alleviated tunnel requirement.

Let N be the set of n AID stations. Each $x \in N$ connects with a subset N_x of k randomly-selected stations via IPSec tunnels. Consider an arbitrary station $y \in N$. Let P_l be the probability for x to reach y in l or less hops, and E_l be the expected number of stations that can be reached by x in l or less hops. First we derive P_2 and E_2.

1. The probability of $y \neq x$ is $p_0 = \frac{n-1}{n}$.
2. If $y \neq x$, the conditional probability for x not reaching y by one hop is $p_1 = \frac{n-k-1}{n-1}$.
3. If $y \neq x$ and x cannot reach y by one hop, the conditional probability for x not reaching y by two hops is $p_2 = (\frac{n-k-1}{n-2})^k$, which is explained as follows: Consider an arbitrary station $z \in N_x$. Besides x and z, there exists $n-2$ nodes. Among them, $k-1$ stations are z's neighbors, connecting via tunnels. The probability of y not being a neighbor is $\frac{n-2-(k-1)}{n-2} = \frac{n-k-1}{n-2}$. There are k nodes in N_x. The probability for none of them being y's neighbor is $(\frac{n-k-1}{n-2})^k$

In summary, we have

$$\begin{aligned} P_2 &= 1 - p_0 p_1 p_2 = 1 - \frac{(n-k-1)^{k+1}}{n(n-2)^k} \\ E_2 &= n(1 - p_0 p_1 p_2) = n - \frac{(n-k-1)^{k+1}}{(n-2)^k} \end{aligned} \tag{1}$$

Theorem 1 *If $n > 2$ and $k = \sqrt{n}$, then $P_2 > 1 - \frac{1}{e}$ and $E_2 > n(1 - \frac{1}{e})$.*

The proof is omitted. Similarly, if x cannot reach y in two or less hops, the conditional probability for x not reaching y by three hops is $p_3 \approx (\frac{n-k-1}{n-2})^{E_2-k-1}$. Hence, we have

$$\begin{aligned} P_3 &= 1 - p_0 p_1 p_2 p_3 \approx 1 - \frac{(n-k-1)^{E_2}}{n(n-2)^{E_2-1}} \\ E_3 &= n(1 - p_0 p_1 p_2 p_3) \approx n - \frac{(n-k-1)^{E_2}}{(n-2)^{E_2-1}} \end{aligned} \tag{2}$$

Let $E_0 = 1$. By induction, for $l \geq 1$,

$$\begin{aligned} P_l &\approx 1 - \frac{(n-k-1)^{E_{l-1}}}{n(n-2)^{E_{l-1}-1}} \\ E_l &\approx n - \frac{(n-k-1)^{E_{l-1}}}{(n-2)^{E_{l-1}-1}} \end{aligned} \tag{3}$$

Let $k = 100$, i.e., each station has 100 neighbors. Even when n is as large as 10,000, $P_3 = 1 - 3.95 \times 10^{-28}$, which means almost certainly a) that the AID stations are fully connected and b) that the shortest routing distance between any two nodes does not exceed 3 hops. For all practical purposes, a modest value of k is sufficient to ensure a connected overlay network with small diameter.

The random overlay network can be constructed by an enhanced VPN tool [2] that constructs the random connectivity, generates the device configurations, and deploys the configurations to individual devices to establish the IPSec tunnels. The tool must make sure every pair of nodes is at most three hops away. Otherwise, the random connectivity has to be regenerated, though that is extremely unlikely to happen even for a modest k as we discussed above.

2.3 Constructing Tunnel Tree from Client Networks to Server under Attack

When a server s is under a DoS attack, the edge device detects the attack based on a preconfigured policy, e.g., the rate of connections exceed 10,000 per second or the traffic volume exceeds 10 Megabits per second. The device then signals its AID station, denoted as A_s. To inform all AID stations about the attack, we adopt a hybrid push-n-pull model. Suppose $k = \sqrt{n}$. Let π be the type of client traffic to be protected, which can be "SYN segments only", "all IP packets", etc. Let i-neighborhood be the set of stations at most i hops away. The broadcast in the i-neighborhood can be easily implemented by augmenting the message with a TTL field whose initial value is i. A_s performs a 2-neighborhood broadcast by sending an ALERT(s, π) message to its neighbors, which in turn forwards the message to their neighbors except A_s. By Theorem 1, this "push" phase will reach a majority of the AID stations with k^2 ($= n$) messages. Note that 3-neighborhood broadcast would be too expensive.

Upon receipt of the first ALERT, a station x informs all its slave processes to update their crypto-ACLs to admit the client traffic of type π and destination s into the tunnels. Suppose x receives the ALERT from y. x should also update the crypto-ACL of the tunnel between itself and y, so that the tunneled traffic received from the clients can be further tunneled to y, which is one hop closer to the server. Each station in the 2-neighborhood keeps the number of tunnel hops from s, which can be easily determined by the TTL field carried by ALERT.

For an AID station that is not reached by the "push" phase, it joins the defense by "pulling" for a path to A_s. Each AID station x periodically sends a PULL message to q randomly-selected neighbors. Let Π_s be the set of stations that are within two hops from A_s. $|\Pi_s| = E_2 > n(1 - \frac{1}{e})$ by Theorem 1. The probability for the pulling to locate at least one station in Π_s is

$$1-(\frac{n-1-|\Pi_s|}{n-1})^q > 1-(1-\frac{n(1-\frac{1}{e})}{n-1})^q > 1-\frac{1}{e^q} \quad (4)$$

If $q = 10$, the probability is greater than 0.99995, which will almost certainly happen. Let the pulling period be T_p. When a station receives PULL, it forwards all ALERT messages received during a recent period, which should be greater than T_p in order to accommodate the 2-neighborhood broadcast delay. It can be trivially proved that, if a station does not received an ALERT in the push phase, it will receive the ALERT by pulling with a probability given by (4).

The push-n-pull process establishes a tunnel tree from the registered client networks to the server under attack. The depth of the tree is three hops, with the majority of stations within two hops from the server. A station y is another station x's parent in the tree if x receives its first ALERT from y. It is straightforward for every station to keep track of which is the parent and which are the children.

2.4 Distributed Virtual-Clock Packet Scheduling

For each server s under attack, a separate tunnel tree rooted at A_s is dynamically constructed. The traffic in different tunnel trees is processed independently. Our discussion will focus on a single tunnel tree.

The control traffic flows downstream from the root A_s to the leaf stations, and the data traffic flows upstream from the client networks to A_s and then to s. Each AID station receives data packets from downstream tunnels and multiplexes them into the upstream tunnel towards A_s. A virtual client puzzle is simulated by spacing the data packets appropriately. A_s periodically broadcasts a CONTROL(T) message in the tree, where T is called the *waiting interval*. Each AID station x maintains a virtual clock VC_u [9] (initialized to be the local system clock) for every tunnel u connecting with a client network. Assume the system clocks of all AID stations are well synchronized.

1. When x receives a packet from a tunnel u, it updates VC_u as follows.

$$VC_u = \max\{VC_u,\ current\_time\} + T \times L \quad (5)$$

   where L is the length of the packet. x then labels the packet with a timestamp equal to VC_u.

2. When x receives a packet from a tunnel connecting with another AID station, the packet will already have a timestamp.

All received packets are sorted in a heap array in the ascending order based on their timestamps. A packet is scheduled for entering the upstream tunnel only when the timestamp of a packet does not exceed the current system clock by α, which is a constant larger than the accumulated clock skew on any branch of the tunnel tree.

The timestamp-based scheduling performed by all AID stations in the tree collectively ensures that each client network receives a fair share of the server's capacity and the unused bandwidth left by some clients are evenly spread among the other clients. Each client network is able to send data to the server at a rate of up to $\frac{1}{T}$. If a registered client network hosts an attack source, the high volume of attack traffic will quickly advance its virtual clock, which results in large timestamps for its packets. The packets will be placed to the end of the heap and dropped if the maximum size of the heap is reached.

The server receives data from registered clients via A_s and from unregistered clients via the Internet. The server's

edge router reserves a portion of the server's capacity, denoted as c_s, for traffic from A_s based on a service contract. Consequently, the attack traffic from the Internet does not affect the registered clients. A_s starts by broadcasting CONTROL(T) with $T = \frac{1}{c_s}$ in the tunnel tree. If the attack traffic comes from some registered clients, the traffic rate received by A_s will exceed c_s. In this case, A_s broadcasts CONTROL messages periodically with new values of T. First an exponential phase is used to update T by doubling it after each period (e.g., 10 seconds). Doubling T makes the virtual clocks run twice as fast, which effectively reduces the maximum rate per client network by half. Once the arrival traffic rate is below c_s, a linear phase starts. Suppose T is changed from I to $2I$ by the last update of the exponential phase. The linear phase will decrease T by εI periodically until the arrival rate is above c_s, at which moment we call the system converges.

Let T_1 and T_2 be the waiting intervals before and after the last update of the linear phase, respectively. $T_2 = T_1 - \varepsilon I \leq (1-\varepsilon)T_1$. Hence, $\frac{1}{T_2} \geq \frac{1}{1-\varepsilon}\frac{1}{T_1}$, where $\frac{1}{T_1}$ and $\frac{1}{T_2}$ are the traffic rates allowed from each registered client network before and after the last update, respectively. Therefore, the total arrival rate at A_s is improved at most by a factor of $\frac{1}{1-\varepsilon}$. Because the arrival rate is below c_s before the update, it must be below $\frac{1}{1-\varepsilon}c_s$ after the update. We require that the server's capacity should be at least $\frac{1}{1-\varepsilon}c_s$ and the client traffic from AID is given higher priority than that from the Internet.

After converging, the updates of T may be restarted if the arrival rate goes beyond the range of $[c_s, \frac{1}{1-\varepsilon}c_s]$.

3 Conclusion

The traditional anti-DoS strategy is mainly along the line of modifying the OS, protocol stacks, or other software of end hosts or routers. The resulting defense systems often require wide-spread deployment. In this paper, we take a new defense strategy to provide a global anti-DoS service (AID) for general-purpose TCP-based public servers. The service is open to all clients and does not need authentication. It requires no change to the end systems or the routing infrastructure. It works with the existing applications and takes action only when a DoS attach is undergoing. The main technical contribution is a new random overlay network, which is the key to the system's scalability and efficiency. While AID appears to be a promising direction, we recognize that there are many issues to be further addressed in the future, for example, how to protect the UDP traffic, how to resist the compromise of some AID stations, and how to perform traceback within the AID system.

References

[1] P. Ferguson and D. Senie. Network Ingress Filtering: Defeating Denial of Service Attacks Which Employ IP Source Address Spoofing. *IETF, RFC 2267*, Janurary 1998.

[2] S. Hinrichs and S. Chen. Network Management Based on Policies. *SPIE Multimedia Computing and Networking Conference*, January 2000.

[3] A. D. Keromytis, V. Misra, and D. Rubenstein. SOS: Secure Overlay Services. *Proc. of ACM SIGCOMM'2002*, August 2002.

[4] D. Moore, G. Voelker, and S. Savage. Inferring Internet Denial of Service Activity. *Proc. of USENIX Security Symposium'2001*, August 2001.

[5] K. Park and H. Lee. On the Effectiveness of Route-Based Packet Filtering for Distributed DoS Attack Prevention in Power-Law Internets. *Proc. of ACM SIGCOMM'2001*, August 2001.

[6] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications . *ACM SIGCOMM'2001*, 2001.

[7] C. Systems. Cisco PIX 500 Series Firewalls. *http://www.cisco.com/warp/public/cc/pd/fw/sqfw500/*.

[8] H. Wang, D. Zhang, and K. G. Shin. SYN-dog: Sniffing SYN Flooding Sources. *Proc. of 22 nd International Conference on Distributed Computing Systems (ICDCS'02)*, July 2002.

[9] L. Zhang. VirtualClock: A New Traffic Control Algorithm for Packet Switching Networks. *ACM Transactions on Computer Systems*, 9(2):101–124, May 1991.