Mayday: Distributed Filtering for Internet Services

David G. Andersen
MIT Laboratory for Computer Science
dga@nms.lcs.mit.edu

Abstract

Mayday is an architecture that combines overlay networks with lightweight packet filtering to defend against denial of service attacks. The overlay nodes perform client authentication and protocol verification, and then relay the requests to a protected server. The server is protected from outside attack by simple packet filtering rules that can be efficiently deployed even in backbone routers.

Mayday generalizes earlier work on Secure Overlay Services. Mayday improves upon this prior work by separating the overlay routing and the filtering, and providing a more powerful set of choices for each. Through this generalization, Mayday supports several different schemes that provide different balances of security and performance, and supports mechanisms that achieve better security or better performance than earlier systems. To evaluate both Mayday and previous work, we present several practical attacks, two of them novel, that are effective against filtering-based systems.

1 Introduction

Denial of service (DoS) attacks are potentially devastating to the victim and require little technical sophistication or risk exposure on the part of the attacker. These attacks typically attempt to flood a target with traffic to waste network bandwidth or server resources. To obtain the network bandwidth necessary to attack well-connected Internet services, attackers often launch Distributed DoS (DDoS) attacks, where tens to thousands of hosts concurrently direct traffic at a target. The frequency of these attacks is startling—one analysis of attack "backscatter" suggests that hundreds of these attacks take place each day [22]. DDoS attacks no longer require a high degree of sophistication. So-called "rootkits" are available in binary form for a variety of platforms, and can be deployed using the latest off-the-shelf exploits. Even worm programs have been used to launch DDoS attacks [9].

While technical measures have been developed to prevent [14, 23, 17] and trace [30, 12, 27] DDoS attacks, most of these measures require wide-spread adoption to be successful. Unfortunately, even the simplest of these measures, filtering to prevent IP address spoofing, is not globally deployed despite years of advocacy. While there is some interim benefit from the incremental deployment of earlier measures, they lack the deployment incentive of a solution that provides immediate relief to the deployer.

An ideal DDoS prevention system stops attacks as close to their source as possible. Unfortunately, the targets of attacks have the most incentive to deploy solutions, and deployment is easiest inside one's own network. Intrusive systems that perform rate-limiting or that require router modifications hold promise, but most Internet Service Providers (ISPs) are unwilling or unable to deploy these solutions in the places where they would be most effective—in their core or at their borders with other ISPs.

We study a set of solutions that are more resource-intensive to deploy, because they require overlay nodes, but that are easily understood and implemented by ISPs using conventional routers. Trace-based *reactive* solutions impose no overhead during normal operation, but suffer from a time lag before recovering from an attack. Our solution, an architecture called Mayday, provides *pro-active* protection against DDoS attacks, imposing overhead on all transactions to actively prevent attacks from reaching the server. Mayday generalizes the Secure Overlay Services (SOS) approach [20]. Mayday

This research was sponsored by the Defense Advanced Research Projects Agency (DARPA) and the Space and Naval Warfare Systems Center, San Diego, under contract N66001-00-1-8933. David G. Andersen is supported by a Microsoft Graduate Fellowship.

uses a distributed set of **overlay nodes** that are trusted (or semi-trusted) to distinguish legitimate traffic from attack traffic. To protect a server from DDoS traffic, Mayday prevents general Internet hosts from communicating directly with the server by imposing a router-based, network-layer **filter ring** around the server. Instead, clients communicate with the overlay nodes, who verify that the client is permitted to use the service. These overlay nodes then use an easily implemented **lightweight authenticator**, such as sending the traffic to the correct TCP port on the server, to get through the filter ring. Within this framework, SOS represents a particular choice of authenticator and overlay routing, using distributed hash table lookups to route between overlay nodes, and using the source address of the overlay node as the authenticator. We explore how different organizations of the authentication agents operate under various threat models, and present several lightweight authenticators that provide improved levels of defense over source address authentication.

Finally, we define several threat models with which we evaluate pro-active DDoS protection. Within these threat models, we present several attacks, including a novel scanning method we call next-hop scanning, that are effective against SOS, certain variants of Mayday, and against conventional router-based filtering of DDoS attacks.

2 Related Work

DoS flooding attacks have been well studied in the recent literature. Most work in this area has been aimed at either preventing attacks by filtering, or at detecting attacks and tracing them back to their origin. Overlay networks have been used in many contexts to speed deployment of new protocols and new functionality.

2.1 Attack Prevention

The most basic defense against anonymous DoS attacks is *ingress filtering* [13]. Ingress filtering is increasingly deployed at the edge of the network, but its deployment is limited by router resources and operator resources. Ingress filtering also interferes with Mobile IP techniques and split communication systems such as unidirectional satellite systems. Despite these limitations, in time, address filtering should become widespread, enhanced by mechanisms such as Cisco's Reverse Path Filtering. However, ingress filtering is most effective at the edge; deployment in the core, even if it becomes technically feasible, is not completely effective [23].

Mazu Networks [21] and Arbor Networks [6] provide DoS detection and prevention by creating models of "normal" traffic and detecting traffic that violates the model. If an attack is detected, Mazu's tools suggest access lists for routers. If the Mazu box is installed in-line with the network, it can shape traffic to enforce a previously good model. Asta Networks' Vantage analyzes NetFlow data to detect DoS attacks on high-speed links and suggest access lists to staunch the flood [7]. These access lists must be deployed manually, and provide reactive, not proactive, assistance to the victim of a DoS attack. Because these schemes result in access lists being applied at routers, many of the probing attacks we discuss in Section 4 can be used against these solutions as well.

Pushback provides a mechanism for pushing rate-limiting filters to the edges of an ISP's network [17]. If attack packets can be distinguished from legitimate traffic (as in the case of a SYN flood), Pushback's mechanisms can effectively control a DoS attack. In the general case, Pushback will also rate-limit valid traffic. If the source of the traffic is widely distributed over the network, Pushback is less effective. In any event, Pushback is effective at reducing collateral damage to other clients and servers that share links with the DoS target, but this scheme requires new capabilities of routers, slowing deployment.

2.2 Attack Detection

ICMP traceback messages were proposed as a first way of tracing the origins of packets [8]. Under this scheme, routers would periodically send an ICMP message to the destination of a packet. This message would tell the recipient the link on which the packet arrived and left, allowing the recipient of a sufficient quantity of ICMP traceback messages to determine the path taken by the packets.

To avoid out-of-band notifications, Savage et al. use probabilistic inline packet marking to allow victims to trace attack packets back to their source [27]. In this scheme, routers occasionally note in the packet the link the packet has traversed; after sufficient packets have been received by the victim host, it can reconstruct the full path taken by the packets. Dean et al., treat the path reconstruction problem as an algebraic coding problem [12]. These refinements improve the performance and robustness of the packet marking, but the underlying technique is similar to the original.

The probabilistic traceback schemes require that a large amount of data be received by a victim before path reconstruction can be performed. To allow traceback of even a single packet, the Source Path Isolation Engine

(SPIE) system records the path taken by *every* packet that flows through a router [30]. SPIE uses a dense bloom-filter encoding to store this data efficiently and at high speeds. While it provides exceptional flexibility, SPIE requires extensive hardware support.

2.3 Overlay Networks

Overlay networks have long been used to deploy new features. Most relevant to this work are those projects that used overlays to provide improved performance or reliability. The Detour [26] study noted that re-routing packets between hosts could often provide better loss, latency, and throughput than the direct Internet path. The RON project experimentally confirmed the Detour observations, and showed that an overlay network that performs its own network measurements can provide improved reliability [4].

Content Delivery Networks such as Akamai [3], and Cisco's Overcast [18] use overlay networks to provide faster service to clients by caching or eliminating redundant data transmission. The ideas behind these networks would integrate well with Mayday; in fact, the Akamai network of a few thousand distributed nodes seems like an ideal environment in which to deploy a Mayday-like system.

Mixnet-based anonymizing overlays like Tarzan [15] are designed to prevent observers from determining the identity of communicating hosts. The principles used in these overlays, primarily Chaumian Mixnets [10], can be directly used in a system such as Mayday to provide greater protection against certain adversaries. We discuss this further in Section 3.5.

3 Design

The design of Mayday evolved from one question: Using existing network capabilities, how do we protect a server from DDoS attacks while ensuring that legitimate clients can still use the services it provides? To answer this question, we restricted ourselves to using only routers with limited packet filtering abilities, or more powerful hosts that aren't on the forwarding path. We wish to provide protection against realistic attackers who control tens or thousands of machines, not malicious network operators or governments. Before exploring the design of our system, we define these attackers and the capabilities they possess. For this discussion, the *server* is a centralized resource that is required in order to provide some service. *Clients* are authorized to use the service, but are not trusted to communicate directly with the server because clients are more numerous and more prone to compromise. *Overlay nodes* are hosts scattered around the Internet that act as intermediaries between the clients and the server.

3.1 Attacker Capabilities

DDoS attacks can be mounted with a relatively low degree of technical sophistication. We focus exclusively on *flooding* attacks, and not on attacks that could crash services with incorrect data. (Using the overlay nodes as protocol verifying agents could prevent some data-based attacks as well.) The simplest flooding attacks (which may be effective if launched from a well-connected site) require only a single command such as `ping`. Many sophisticated attacks come pre-packaged with installation scripts and detailed instructions, and can often be used by people who may not even know how to program. The greatest threat to many Internet services comes from relatively simple attacks because of their ease of use and ready accessibility. We therefore concentrate on simpler attacks.

We assume that all attackers can send a large amount of data in arbitrary formats from forged source addresses of their choice. Ingress filtering may reduce the number of hosts with this capability, but is unlikely to eliminate all of them. Certain attackers may have more resources available, may be able to sniff traffic at points in the network, and may even be able to compromise overlay nodes. We consider the following classes of attackers:

The **Client Eavesdropper** can view the traffic going to and from one or more clients, but cannot see traffic that has reached an overlay node or the target.

The **Legitimate Client Attacker** is authorized to use the service, or is in control of an authorized client.

The **Random Eavesdropper** can monitor the traffic going to one or more overlay nodes, but cannot choose which overlay nodes are watched.

The **Targeted Eavesdropper** can view the traffic going to and from any particular overlay node, but not all overlay nodes at once (i.e., changing monitored nodes requires non-negligible time).

The **Random Compromise Attacker** can compromise one or more randomly chosen overlay nodes.

The **Targeted Compromise Attacker** can select a particular overlay node, or a series of them, and obtain full control of the node.

We ignore certain attackers. For instance, an attacker capable of watching all traffic in the network, or compromising all nodes concurrently, is too powerful for our model to resist. The difference between these global attackers and the targeted eavesdropper or compromiser is one of time and effort. Given sufficient time, the targeted compromise attacker may be able to control all nodes, but during this time the service provider may counteract the offense.

3.2 Mayday Architecture

The Mayday architecture assumes that some entity—perhaps the server's ISP—has routers around the server that provide its Internet connectivity, and is willing to perform some filtering at those routers on behalf of its client. We term this set of routers the *filter ring*. While the ring could be implemented by filtering at the router closest to the server, this would provide little attack protection, because the traffic would consume the limited bandwidth close to the server. Pushing the filter ring too far from the server increases the chance that nodes inside the filter can be compromised. Instead, the ring is best implemented near the core-edge boundary, where all traffic to the server passes through at least one filtering router, but before the network bottlenecks become vulnerable to attack. To provide effective protection against large attacks, this filtering must be lightweight enough to be implemented in high-speed core routers.

The requirement for a fast router implementation rules out certain design choices. One obvious mechanism would be for clients to use IPSec to authenticate themselves to a router in the filter ring, at which point the router would pass the client's traffic through. If a service provider is capable of providing this service, along with rate limiting, a server should be well-protected from DoS attacks.

The Mayday architecture is designed to work with more limited routers. Modern routers can perform routing lookups very quickly, and many (but not all) can perform a *few* packet filtering operations at line speed. Clients, however, may be many in number, or the set of clients may change dynamically. Client verification may involve database lookups or other heavyweight mechanisms. Access lists in core routers are updated via router configuration changes, so network operators are not likely to favor a solution that requires frequent updates. Creating an access list of authorized clients is probably not practical due to client mobility and the sheer size of such a list; we need filter keys that change less often. We term these filter keys the *lightweight authenticators.*

To handle the joint requirements of client authentication and feasible implementation, we add a fourth type of party, the *overlay nodes*. Clients talk directly to an overlay node, the *ingress node*, not to the server or filter ring. Some of the overlay nodes, the *egress nodes*, can talk to

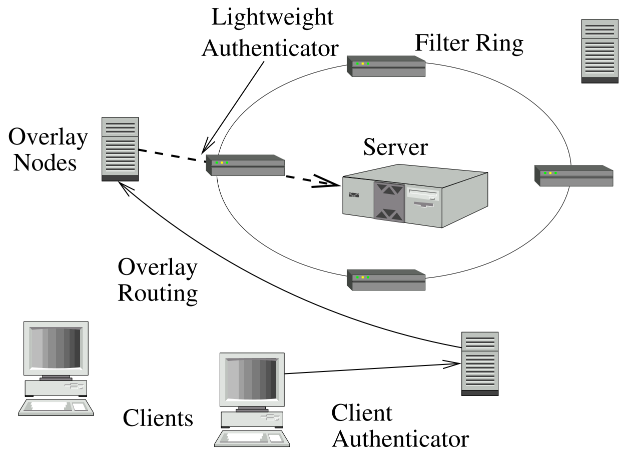

Figure 1: The Mayday architecture. Clients communicate with overlay nodes using an application-defined client authenticator. Overlay nodes authenticate the clients and perform protocol verification, and then relay requests through the filter ring using a lightweight authenticator. The server handles requests once they pass through the network-layer filter ring.

the server through the filter ring. If the ingress node is not also an egress node, the request must be routed through the overlay to an egress node. Figure 1 shows the general Mayday architecture.

Using this architecture, a designer can make several choices to trade off security, performance, and ease of deployment. First, the designer can first pick one of several overlay routing methods: more secure overlay routing techniques reduce the impact of compromised overlay nodes, but increase request latency. Second, the designer can pick one of several lightweight authenticators, such as source address or UDP/TCP port number. The choice of authenticator affects both security and the overlay routing techniques that can be used. The security and performance of the resulting system depend on the *combination* of authenticator and overlay routing. We discuss the properties of these combinations in Section 3.6 after describing the individual mechanisms.

3.3 Client Authentication

Clients must authenticate themselves to the overlay before they are allowed to access the server. The nature of the client authentication depends on the service being protected. If Mayday is used to protect a small, private service, clients could be authenticated using strong cryptographic verification. In contrast, if Mayday is protecting a large, public service such as Yahoo!, client authentication may be only a database verification of the user's password. Mayday leaves client authentication up to the system designer, since it is inextricably linked to the specific application being protected.

3.4 Lightweight Authenticators

Mayday uses lightweight authentication tokens to validate communication between the overlay node(s) and the server. Mayday requires its tokens be supported with low overhead by commodity routers. Modern routers can filter on a variety of elements in the packet header, such as source and destination address, UDP or TCP port number, and so on. Several of these fields can be used as authenticators. All "source" addresses and ports refer to the egress node; "destination" addresses and ports refer to the server. Each of these fields has its own strengths and weaknesses as a lightweight authenticator:

- **Egress Source Address**: Source filtering is well understood by network operators, and gains effectiveness when other providers deploy IP spoofing prevention. It limits the number of overlay nodes that can communicate with the server. SOS uses this authenticator.

- **Server Destination Port**: The UDP or TCP destination port is an obvious key to use. If the overlay network has fewer than 65,000 nodes, this key provides a larger space in which an attacker must search to get through the firewall. It allows multiple authorized sources to communicate with the server. In other respects, it is similar to source address authentication. The source port can also be used, but this limits the total number of concurrent connections to the server.

- **Server Destination Address**: If the server has a variety of addresses that it can use, the destination address can be used as an authentication token. For example, if a server is allocated the netblock 192.168.0.0/24, its ISP would announce this entire block to the world. Internally, the server would only announce a single IP address to its ISP, and send a null route for the remaining addresses. Thus, a packet to the correct IP would go to the server, but packets to the other IP addresses would be dropped at the border routers. The advantage of this mechanism is that it requires no active support from the ISP to change filters, and uses the fast routing mechanisms in routers, instead of possibly slower filtering mechanisms[1]. Because it uses standard routing mechanisms, updates could be pushed out much more rapidly than router reconfigurations for filter changes. We term this effect *agility*, and discuss later how it can provide freshness to authenticators. The disadvantage is that it wastes address space (a problem solved by IPv6, though IPv6 has its own deployment delays). Destination address filtering is unique in that it can be changed very dynamically by routing updates, even in a large network of routers.

- **Other header fields**: Some routers can filter on attributes like protocol, packet size, and fragment offset. Each of these fields can be manipulated by the egress node to act as a lightweight authenticator, but they require lower-level hacks to set. While they could be useful for providing additional bits of key space, they are less usable than port or address filtering, except to provide some security through obscurity.

- **User-defined fields**: Firewalls can filter on user-defined fields inside packets. This approach provides a huge keyspace and source address flexibility, but few core routers support this feature.

Authentication tokens can be combined. Using both source address and port verification provides a stronger authenticator than source address alone, making some of the attacks we discuss in section 4 difficult to pull off.

3.5 Overlay Routing

The choice of overlay routing can reduce the number of overlay nodes that have direct access to the server, thus providing increased security. These choices can range from direct routing, in which every overlay node can directly access the server (i.e. be an egress node), to Mixnet-style routing ("onion routing"), in which the other overlay nodes do not know which is the egress node [15].

The choice of lightweight authenticator affects which overlay routing techniques can be used. For instance, using source address authentication with proximity routing is extremely weak, because an attacker already knows the IP addresses of the overlay nodes, and any of those addresses can pass the filter.

- **Proximity Routing**: By picking the overlay node nearest the client (similar to Akamai and other CDNs [3]) or the node that provides the best performance between client and server [4, 18], the system can provide high performance with low

[1] An interesting manual use of destination filtering occurred during the Code Red worm in 2001. The worm was designed to flood the IP address of `www.whitehouse.gov`, but had a hardcoded address, not a DNS lookup. The site administrators changed the service address and filtered the old address to successfully protect themselves from the attack.

overhead. In fact, when combined with overlay-level caching, this design could provide better performance than direct client-server communication. Proximity routing requires that all overlay nodes possess the lightweight authenticator.

- **Singly-Indirect Routing**: The ingress node passes the message directly to the egress node, which sends the message to the server. All overlay nodes know the identity of the egress node.
- **Doubly-Indirect Routing**: Ingress nodes send all requests to one or more overlay nodes, who then pass the traffic to the egress node. Only a subset of overlay nodes know the identity of the egress node. SOS uses this scheme.
- **Random Routing**: The message is propagated randomly through the overlay until it reaches a node that knows the lightweight authenticator. Adds $O(N)$ additional overlay hops, but provides better compromise containment. In most ways, this routing is inferior to mix routing.
- **Mix Routing**: Based on Mixnets [10] and the Tarzan [15] anonymous overlay system. A small set of egress nodes configure encrypted forwarding tunnels through the other overlay nodes in a manner such that each node knows only the next hop to which it should forward packets, not the ultimate destination of the traffic. At the extreme end of this style, *cover traffic*—additional, fake traffic between overlay nodes—can be added to make it difficult to determine where traffic is actually originating and going. This difficulty provides protection against even the targeted eavesdropper and compromise attacker, but it requires many overlay hops and potentially expensive cover traffic.

3.6 Choice of Authenticator and Routing

The major question for implementation is which combination of overlay routing and authenticator to use. An obvious first concern is practicality: If a service provider is only able to provide a certain type of filtering, the designer's choices are limited. There are three axes on which to evaluate the remaining choices: performance, security, and agility. Many combinations of authenticator and routing fall into a few "best" equivalence classes that trade off security or performance; the remaining choices provide less security with the same performance, or vice versa.

High performance: Proximity routing provides the best performance, but is vulnerable to the random eavesdropper. Works with any authenticator *except* source address, since the address of the overlay nodes is known. Blind DoS attacks against the system are difficult, since all nodes can act as ingress and egress nodes. Singly-indirect routing with source address provides equivalent protection with inferior performance.

Eavesdropping Resistance, Moderate Performance: Singly-indirect routing, when used with any authenticator other than source address, provides resistance to the random eavesdropper and random compromise attack, because only a small number of nodes possess the authentication key.

SOS: The SOS method uses doubly-indirect routing with source address authentication. In the SOS framework, packets enter via an "access node," are routed via a Chord overlay [32] to a "beacon" node, and are sent from the "beacon" node to the "servlet" node. The servlet passes packets to the server. This method provides equivalent security to the singly-indirect scheme above, but imposes at least one additional overlay hop.

Agility: singly-indirect routing with destination address authentication provides an agile (and deployable) system. Because routing updates, not manual configuration changes, are used to change the lightweight authenticator, it is feasible to update the authentication token often. This agility can be used to resist adaptive attacks by changing the authentication token before the attack has sufficiently narrowed in on the token. Destination address authentication can provide this benefit in concert with other authenticators (such as port number) to provide an agile scheme with a large number of authenticators.

Maximum Security: By using Mix-style routing with cover traffic, a service provider can provide some resistance against the targeted compromise attacker (With 3-hop Tarzan routing, an attacker must compromise 24 nodes to reach the egress node). By using agile destination-address based authentication, the service provider gains resistance to adaptive attacks. By combining the agile authenticator with port number authentication, the system increases its key space, while retaining the ability to recover from egress node failures. Alternately, source address authentication would slow this recovery, but it practically reduces the number of attack nodes that can successfully be used since many Internet hosts are filtered.

3.7 Switchable Protection

By using destination address-based filtering, we can provide *switchable* DoS protection: When no attack is present, clients may directly access the service. When an attack commences, the system can quickly and automatically switch to a more secure mode, assuming that some channel exists to notify the nodes and routers of the change. This allows us to use Mayday as both a reactive and a proactive solution.

The service provider is given two or more IP addresses. IP address A is the "normal" mode address, and the other addresses are the "secure" mode addresses. When an attack commences, the service sends a routing update (in the same manner as destination-address based authentication) to change to one of the secure addresses.

The limiting step in reactive DoS protection is configuring access lists at many routers concurrently. To speed this step, the ISP configures two sets of access lists *in advance*. The first list permits access from all clients (or all overlay nodes, for proximity routing) to the normal mode address A. The second list restricts access with a lightweight authenticator, and applies to the secure mode addresses. The server can quickly switch modes by sending a routing update.

This scheme works best when normal mode is proximity routing through all overlay nodes, and secure mode involves more stringent routing and filtering. In this case, the addresses to which clients connect do not change, and client connections need not be interrupted at the commencement of a DDoS attack. If a brief interruption is tolerable, a DNS update can be pushed out to point new connections to the overlay nodes.

3.8 Changing Authenticators or Overlay Nodes

Changing the authentication key or the overlay nodes through which traffic passes could break currently open connections. Fortunately, the communication between the ingress node and the server is completely under the control of the system designer. When a connection between the ingress node and server is interrupted by address or port changes, the designer can use mechanisms such as TCP Migrate [29] or other end-to-end mobility solutions to transparently reconnect the session. Using these mechanisms between ingress node and server would not require client changes.

4 Attacks and Defenses

The ability of an overlay-based architecture to resist simple flooding attacks was explored in the SOS study. For various simple attack models, a sufficiently large number of overlay nodes can resist surprisingly strong attacks targeted against the server or against individual overlay nodes. In this section, we examine a more sophisticated set of attacks than the simple flooding explored in earlier work.

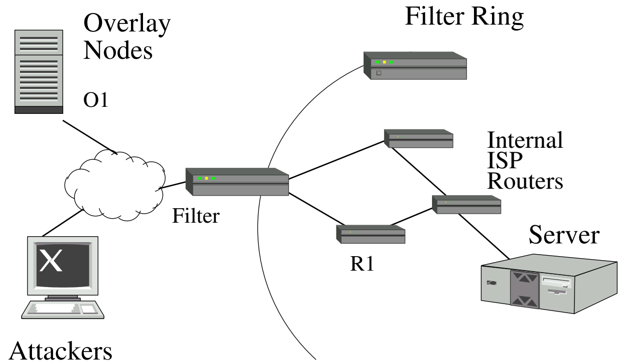

Figure 2: The framework for considering attacks. Overlay nodes and clients are both outside the filter ring. Inside the filter ring may be more ISP routers, which eventually connect to the server.

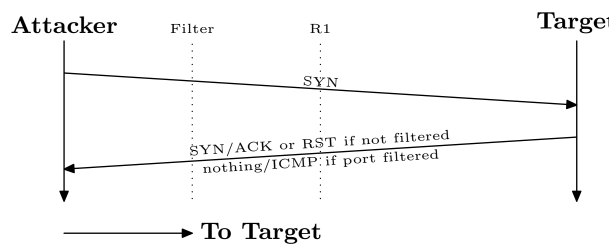

Figure 3: A simple port-scan. The attacker sends packets directly to the target to determine which ports are open.

We view these attacks within the environment shown in Figure 2. We first present several probing attacks that can quickly determine a valid lightweight authenticator to defeat the DDoS protection. We then examine more sophisticated flooding attacks, and examine the effects of eavesdropping and compromise attacks. We assume that attackers may learn the ISP router topology by various means [31, 5] because it is a shared resource.

4.1 Probing

Several lightweight authenticators, such as destination port or destination address, allow arbitrary hosts to communicate directly with the target. While this provides flexibility and higher performance, it can be vulnerable to simple port-scanning attacks (Figure 3). If the target machine will reply to any packet with the lightweight authenticator, it is a trivial matter to scan, say, the 64,000 possible destination ports, or the 256 addresses in a /24 netblock. On a 100 Mbps Ethernet, a full port scan takes about 11 seconds. To prevent these attacks from succeeding, a **secondary key**, drawn from a large keyspace, is

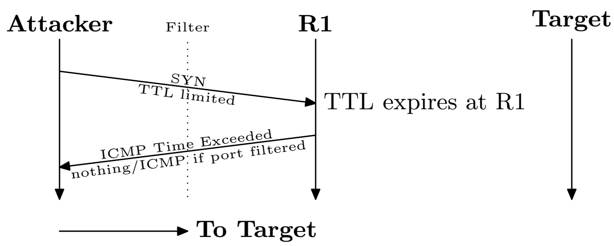

Figure 4: Firewalking. The attacker uses a traceroute-like mechanism to probe the ports that are allowed through the filter ring, without needing replies from the actual target.

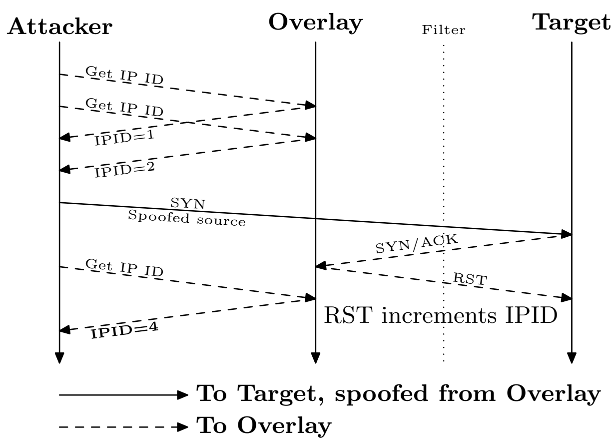

Figure 5: Idlescan indirect probing. The attacker spoofs a TCP SYN packet to the target. If the packet gets through the filter, the target replies with a TCP ACK to the overlay node. The overlay generates a RST because the connection does not exist. The attacker notices the IP ID increment at the overlay node when it sends the RST to determine if the packet got through the filter ring.

needed. While packets with a valid lightweight authenticator will go through the firewall, the server will respond to only packets with the proper secondary key. Clearly, this approach requires considerable attention to detail at the host level for filtering out host responses (ICMP port unreachables or TCP resets). The secondary key could be the addresses of the valid correspondent hosts, a key inside the packets, or heavier weight mechanisms such as IPSec.

The application of the secondary key is complicated by techniques such as *Firewalking* [16] that use Time-To-Live (TTL) tricks to determine what ports a firewall allows through, without requiring that the target host reply to such messages. Figure 4 shows an example of firewalking. Firewalking can be defeated by blocking ICMP TTL exceeded messages at the filter ring, but this breaks utilities like `traceroute` that rely on these messages.

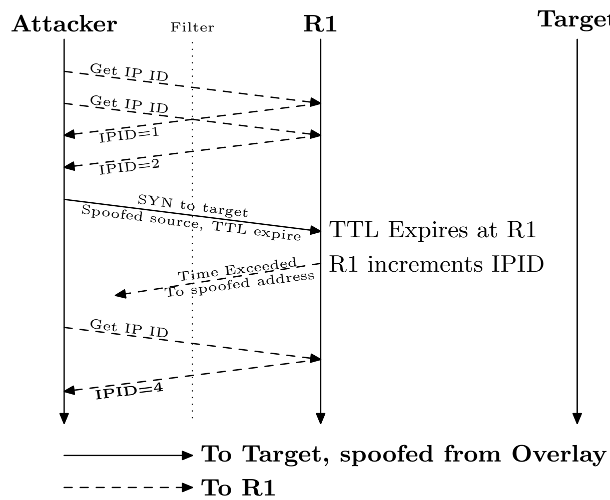

Figure 6: Next-hop scan. This attack combines the idlescan and firewalking to determine from an interior router if packets got through the firewall.

If the filter ring uses source address authentication, attackers can use indirect probing mechanisms to determine the set of source hosts that can reach the server. Tools such as Nmap [33] and Hping [25] can use IP ID increment scanning (or *Idlescanning*) [24] to scan a host indirectly via a third party. Figure 5 shows an idlescan wherein the attacker watches to see if the overlay node has received TCP ACK packets from the target. If it has, it will reply with a TCP RST packet, because it didn't originate a connection to the target. Transmitting this RST causes the overlay node to increment its IP ID, and therefore an attacker can conclude that the probe packet passed the filter by watching the overlay node's IP ID sequences. This technique is limited to TCP, and can be deterred by implementing IP ID randomization techniques on the overlay nodes. This technique also depends on low or predictable traffic volumes on the overlay nodes.

A variant on idlescanning that we call *next-hop* scanning can use other routers behind the filter ring to determine if spoofed packets are getting through. Figure 6 shows next-hop scanning. Like firewalking, next-hop scanning sends a TTL-limited probe at the target, which expires at some interior router $R1$. $R1$ generates an ICMP time exceeded message. Instead of directly receiving this message (if it's filtered or the source address was spoofed), the attacker indirectly observes the generation of the ICMP reply by the IP ID increment at $R1$.

Figure 7 shows a next-hop scan in action. Between sequence 93 and 94, the attacker machine sent 40 TTL-limited probes at the target, causing a large jump in $R1$'s IP ID. This trace was taken using `hping` on a production Cisco router on the Internet; the IP addresses have been

Source	Seq #	**IP ID change**	rtt
192.168.3.1	seq=91	id=+19	76.5 ms
192.168.3.1	seq=92	id=+16	233.4 ms
192.168.3.1	seq=93	id=+14	259.6 ms
192.168.3.1	seq=94	**id=+61**	76.2 ms
192.168.3.1	seq=95	id=+12	76.6 ms
192.168.3.1	seq=96	id=+10	75.5 ms

Figure 7: Next-hop scan showing IP ID increase at the router after the filter. After packet 93, the attacker sent a burst of packets that went through the filter. This scan method can be used to determine if spoofed packets are permitted to go towards a target, but it requires that the attacker be able to communicate with a router on the path after the filter.

obscured.

Other host vulnerabilities can be used in a similar way, but require that the overlay nodes run vulnerable software. Unlike application or specific host vulnerabilities, the IP ID scans are applicable to a wide array of host and router operating systems. They are difficult to defeat in an overlay context because they require either upgrades to the interior routers to prevent next-hop scanning from working, or much more extensive firewalling techniques than may be practical on shared core routers.

4.2 Timing Attacks

In an N-indirect Mayday network in which only certain overlay nodes are allowed to pass traffic to the server, a malicious client may be able to determine the identity of these nodes by timing analysis. Requests sent to an egress overlay node will often process more quickly than requests that must bounce through an extended series of intermediate nodes; in SOS, overlay traversal adds up to a factor of 10 increase in latency. This attack could allow an attacker to determine the identity of the egress node even in a randomly routed overlay. This attack can be mitigated by using multiple egress nodes and always relaying requests to a different egress node.

4.3 Adaptive flooding

This attack is one step up from blindly flooding the target with spoofed IPs. If the attacker can measure the response time of the target, by collusion with a legitimate client or passively monitoring clients, he can launch a more effective attack than pure flooding. The success of a DoS attack is not binary—intermediate levels of attack may simply slow down or otherwise impair the service provided.

Consider a lightweight authenticator whose keyspace has N possible values (all 64,000 TCP ports, or the 1,000 source addresses of overlay nodes). One value allows traffic to reach the target and consume limited resources. The target has a certain unused capacity, its *reserve*, R. The attacker can generate a certain amount of traffic, T. If $T > R$, the attacker uses up the target's resources, and the target's service degrades.

In most DDoS attacks, $T >> R$: the attacker's force is overwhelmingly large. In this case, the attacker can attack with *multiple* authenticators concurrently. If the attacker uses $\frac{N}{2}$ different authenticators, then 50% of the time, one of those authenticators is valid, and $\frac{2T}{N}$ traffic will reach the target. If the service slows down, the attacker knows that the authenticator was in the tested half of the keyspace. By recursively eliminating half of the remaining nodes in a binary-search like progression, the attacker can identify the authenticator in $O(\log N)$ attack rounds. After this, the full ferocity of the attack will penetrate the filter ring. Even intermediate rounds will likely damage the target.

This attack is slowed down by a large keyspace. When the attack power is sufficiently diluted (i.e., $\frac{2T}{N} < R$), the attack must first linearly probe small batches of the keyspace before identifying a range into which the binary search can proceed. Because this attack takes multiple rounds, key agility is effective at reducing the threat by allowing the system to change the key, ensuring that it remains fresh in the face of an attack.

4.4 Request Flood Attacks

Without careful attention to design, the overlay itself can be used by a malicious client to attack the target. An attacker can use the Akamai network, for example, by requesting identical content from many Akamai nodes concurrently. By reading very slowly from the network (or using an extremely small TCP receiver window), the attacker uses very little bandwidth. The caching overlay nodes, however, request the content as quickly as possible from the origin server, causing an overload.

These attacks are fairly easy to trace, and apply more to large, open systems (such as Akamai) than to closed systems with more trusted clients. However, they point out the need for caution when designing a system to improve performance or security, to ensure that the resulting nodes cannot themselves be used to launder or magnify a DoS attack.

4.5 Compromised Overlay Nodes

Controlling an overlay node allows an attacker not only the ability to see source/destination addresses, but to see the actual contents of the information flowing across the network. An attacker knows anything a compromised node knows.

Furthermore, the attacker can now launch internal attacks against the overlay itself. For example, the SOS system uses Chord [32] to perform routing lookups. The Chord system, and similar distributed hash tables, are themselves subject to a variety of attacks [28]. Any other component of the lookup system is similarly a potential source of cascaded compromise when an overlay node is compromised. This observation argues for keeping the overlay routing as simple as possible, unless the complexity results in needed security gains.

Proximity routing and singly-indirect routing can be immediately subverted when nodes are compromised. Doubly-indirect routing provides a degree of resilience to an attacker who compromises a node in a non-repeatable fashion (physical access, local misconfiguration, etc.). Random routing and mix routing can provide increased protection against compromise, but even these techniques will only delay an attacker who exploits a common flaw on the overlay nodes.

4.6 Identifying Attackers

It is possible to reverse the adaptive flooding attack to locate a single compromised node, if the lightweight authenticator can be changed easily. The search operates in an analogous fashion to the adaptive flooding attack: The server distributes key A to half of the nodes, and key B to the other half. When an attack is initiated with key A, the server knows that the attacker has compromised a machine in that half of the nodes. The search can then continue to narrow down the possibly compromised nodes until corrective action can be taken. This response almost certainly requires the agility of destination address authentication.

5 Analysis

Analysis of "backscatter" traffic suggests that more than 30% of observed DDoS SYN-flood or direct ICMP attacks involved 1000 packets per second (pps) or more, and that about 5% of them involved more than 10,000 pps [22]. This study did not observe indirect attacks that can take advantage of traffic amplifiers, and which can achieve even larger attack rates. Fortunately, these indirect attacks can often be stopped using source address authentication: There are no known attacks that can indirectly generate spoofed traffic.

How powerful are these attacks relative to the sites they attack? A T1 line (~ 1.54 Mbps) is likely the smallest access link that would be used by a "critical" service. With full-size packets (typically 1500 bytes), a T1 line can handle just 128 packets per second. The 30th percentile of DoS attacks is nearly an order of magnitude larger than this. A server in a co-location center with a 10 Mbps Ethernet connection can handle about 830 pps, and a 100 Mbps connected server could not withstand the upper 5% of DoS attacks at 10,000 pps.

For a victim on a T1 line, the top 5% of attacks could mount an adaptive flooding attack against a 100 node overlay with source authentication in under 8 rounds: Dividing the 10,000 pps by 50 nodes gives 200 packets per spoofed node per second, more than the T1 can handle. Thus, an attacker can immediately binary search in the egress node space, taking about $\log_2(100)$ rounds.

Many of the IP ID attacks take about 10 packets per attempted key. At 1000 pps, an attacker could discover a destination-port key in about five minutes. In a doubly-indirect overlay using source address authentication (SOS), the attacker could expect to locate the egress node's IP address in about 50 seconds. Using both of these keys, however, would force the attacker to spend nearly 4 days scanning at extremely high packet rates.

Resource consumption attacks, such as SYN floods, can be more destructive at lower packet rates; One study noted that a Linux webserver could handle only up to 500 pps of SYN packets before experiencing performance degradation [11]. SYN packets are also smaller, and are thus easier for an attacker to generate in large volume. By attacking multiple ingress nodes, and attacker could attempt to degrade the availability of the overlay. The top 5% of the attacks, over 10,000 pps, could disable about $\frac{10{,}000}{500} = 20$ overlay nodes. Modern TCP stacks with SYN cookies or compressed TCP state can handle higher packet rates than older systems, but SYN floods still consume more server resources than pure flooding attacks do.

6 Practical Deployment Issues

Could a Mayday system be practically deployed? We believe so. Service providers like Akamai [3] have existing overlay networks that number in the thousands of nodes. Over the last year, router vendors have created products like Juniper's M-series Internet Processor II ASIC that are capable of performing packet filtering at line speed on high-bandwidth links [19]. ISPs have historically been willing to implement filtering to mitigate extremely large

DoS attacks; this willingness was tempered by the inability of their routers to do line-speed filtering. With the deployment of ASIC-assisted filters, ISPs should be able to deploy a few access list entries for major clients.

Mayday is primarily useful for protecting centralized services. Services may use a central server to ease their design, or it may not be economically feasible for a single service to purchase many under-utilized nodes to protect itself from attacks. In these cases, it may be particularly useful to take a service-provider approach, in which multiple clients contract with a single Mayday provider, who provides a shared overlay infrastructure. The service-provider approach helps amortize the cost of the overlay across multiple clients, and provides shared excess capacity to deal with transient load spikes. Protecting clients in this manner allows a larger overlay network, and reduces the number of entities that ISPs must deal with for creating router access lists.

Finally, DDoS protection is only the first line of defense for servers. The objective of Mayday is to prevent flooding attacks from overwhelming servers. In the real world, servers have a host of additional security problems that they must contend with, and interior lines of defense must still be maintained.

7 Conclusion and Future Work

We have presented a general architecture for using efficient router filtering with semi-trusted overlay nodes to provide denial of service resistance to servers. By generalizing from earlier work, we present several novel mechanisms that can provide improved performance with equivalent security. Designers implementing the Mayday architecture gain the flexibility to trade security for performance to better create a system that matches their needs.

To understand how overlay-based DoS protection would work in the real world, we presented several attacks that are effective against many router-based DoS prevention schemes. By providing options for more precise filtering and more agile rule updates, the Mayday architecture can successfully reduce the impact of these attacks.

While the Mayday architecture can provide a practical and effective proactive defense against DoS attacks, much work remains. Current router architectures are vulnerable to probes like our next-hop scan, and correcting these vulnerabilities will take time. Not all services can afford to protect themselves with Mayday, but still require some protection. There have been many proposals for detecting and preventing DoS attacks at the network layer and up, and sorting through the options remains a formidable task.

Acknowledgments

Many thanks to Alex Snoeren for suggesting the title, and being the initial sounding board for many of these ideas. Angelos Keromytis and Dan Rubenstein pointed out several improvements to this paper, and wrote the original SOS paper that inspired this work. Sanjit Biswas suggested the caching overlay node attack, and Mike Freedman helped shape the discussion about Tarzan and cover traffic. I am indebted to Hari Balakrishnan, Frans Kaashoek, Kevin Fu, Robert Morris, and the USITS reviewers for great discussion and feedback.

References

[1]

[2]

[3] Akamai. http://www.akamai.com, 1999.

[4] ANDERSEN, D. G., BALAKRISHNAN, H., KAASHOEK, M. F., AND MORRIS, R. Resilient Overlay Networks. In *Proc. 18th ACM Symposium on Operating Systems Principles (SOSP)* (Banff, Canada, Oct. 2001), pp. 131–145.

[5] ANDERSEN, D. G., FEAMSTER, N., BAUER, S., AND BALAKRISHNAN, H. Topology Inference from BGP Routing Dynamics. In *Proc. ACM SIGCOMM Internet Measurement Workshop* (Marseille, France, Nov. 2002).

[6] ARBOR NETWORKS. Peakflow for enterprises datasheet. `http://arbornetworks.com/up_media/up_files/Pflow_Enter_datasheet2.1.pdf%`, 2002.

[7] ASTA NETWORKS. Convergence of security and network performance (vantage system overview). `http://www.astanetworks.com/products/data_sheets/asta_data_sheet.pdf`, 2002.

[8] BELLOVIN, S. *ICMP Traceback Messages, Internet-Draft, draft-bellovin-itrace-00.txt, Work in Progress*, Mar. 2000.

[9] CAIDA Analysis of Code-Red. `http://www.caida.org/analysis/security/code-red/`, 2002.

[10] CHAUM, D. Untraceable electronic mail, return addresses and digital pseudonyms. *Communications of the ACM 24*, 2 (1981), 84–88.

[11] DARMOHRAY, T., AND OLIVER, R. Hot spares for DoS attacks. *;Login: The Magazine of USENIX and SAGE* (July 2000).

[12] DEAN, D., FRANKLIN, M., AND STUBBLEFIELD, A. An algebraic approach to IP traceback. *Information and System Security 5*, 2 (2002), 119–137.

[13] FERGUSON, P., AND SENIE, D. *Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing*. Internet Engineering Task Force, Jan. 1998. RFC 2267.

[14] FERGUSON, P., AND SENIE, D. *Network Ingress Filtering*. Internet Engineering Task Force, May 2000. Best Current Practice 38, RFC 2827.

[15] FREEDMAN, M. J., AND MORRIS, R. Tarzan: A peer-to-peer anonymizing network layer. In *Proc. 9th ACM Conference on Computer and Communications Security* (Washington, D.C., Nov. 2002).

[16] GOLDSMITH, D., AND SCHIFFMAN, M. Firewalking: A traceroute-like analysis of IP packet responses to determine gateway access control lists. `http://www.packetfactory.net/firewalk/`, 1998.

[17] IOANNIDIS, J., AND BELLOVIN, S. M. Implementing Pushback: Router-Based Defense Against DDoS Attacks. In *Proc. Network and Distributed System Security Symposium (NDSS)* (San Diego, CA, Feb. 2002).

[18] JANNOTTI, J., GIFFORD, D. K., JOHNSON, K. L., KAASHOEK, M. F., AND O'TOOLE JR., J. W. Overcast: Reliable multicasting with an overlay network. In *Proc. 4th USENIX OSDI* (San Diego, CA, Nov. 2000), pp. 197–212.

[19] JUNIPER NETWORKS. M-series Internet Processor II ASIC Frequently Asked Questions. `http://www.juniper.net/solutions/faqs/m-series_ip2.html`.

[20] KEROMYTIS, A. D., MISRA, V., AND RUBENSTEIN, D. SOS: Secure overlay services. In *Proc. ACM SIGCOMM* [2], pp. 61–72.

[21] Mazu networks. `http://www.mazunetworks.com/solutions/`, 2002.

[22] MOORE, D., VOELKER, G., AND SAVAGE, S. Inferring Internet denial of service activity. In *Proc. USENIX Security Symposium* (Aug. 2001).

[23] PARK, K., AND LEE, H. On the effectiveness of route-based packet filtering for distributed DoS attack prevention in power-law Internets. In *Proc. ACM SIGCOMM* [1].

[24] SANFILIPPO, S. Posting about the IP ID reverse scan. `http://www.kyuzz.org/antirez/papers/dumbscan.html`, 1998.

[25] SANFILIPPO, S. hping home page. `http://www.hping.org/`, 2000.

[26] SAVAGE, S., COLLINS, A., HOFFMAN, E., SNELL, J., AND ANDERSON, T. The End-to-End Effects of Internet Path Selection. In *Proc. ACM SIGCOMM* (Cambridge, MA, Sept. 1999), pp. 289–299.

[27] SAVAGE, S., WETHERALL, D., KARLIN, A., AND ANDERSON, T. Network support for IP traceback. *IEEE/ACM Transactions on Networking 9*, 3 (June 2001).

[28] SIT, E., AND MORRIS, R. Security considerations for peer-to-peer distributed hash tables. In *Proc. 1st International Workshop on Peer-to-Peer Systems (IPTPS)* (Cambridge, MA, Mar. 2002).

[29] SNOEREN, A. C., AND BALAKRISHNAN, H. An end-to-end approach to host mobility. In *Proc. ACM Mobicom* (Boston, MA, Aug. 2000), pp. 155–166.

[30] SNOEREN, A. C., PARTRIDGE, C., SANCHEZ, L. A., JONES, C. E., TCHAKOUNTIO, F., SCHWARTZ, B., KENT, S. T., AND STRAYER, W. T. Single-packet IP traceback. *IEEE/ACM Transactions on Networking 10*, 6 (Dec. 2002).

[31] SPRING, N., MAHAJAN, R., AND WETHERALL, D. Measuring ISP topologies with Rocketfuel. In *Proc. ACM SIGCOMM* [2].

[32] STOICA, I., MORRIS, R., KARGER, D., KAASHOEK, M. F., AND BALAKRISHNAN, H. Chord: A scalable peer-to-peer lookup service for Internet applications. In *Proc. ACM SIGCOMM* [1].

[33] VASKOVICH, F. Nmap stealth port scanner. `http://www.insecure.org/nmap/index.html`, 2002.