# Firebreak: An IP Perimeter Defense Architecture

Paul Francis
Cornell University
Ithaca, NY USA
francis@cs.cornell.edu

**FORWARD:** This paper was submitted as is to (and rejected by) Hotnets '04, and then never submitted anywhere else. I stopped working on the idea when I came to realize that the model being put in place by major ISPs, based for instance on the Riverhead/Cisco, is quite similar to the Firebreak model. The main difference is that Firebreak is cross-ISP, and has a different control model, but I doubt that this difference is really compelling in practice.

## ABSTRACT

After many years of research, the Distributed Denial of Service (DDoS) problem remains essentially unsolved, both in industry and in research. Industry solutions rely primarily on beefing up the bandwidth near the attack target, and/or on intercepting traffic at proxies while keeping the target IP address secret. The former approach is expensive, and the latter amounts to "security through obscurity". Existing research solutions, on the other hand, have so far proven economically infeasible. We propose an architecture, called the firebreak, based on IP-level indirection. With firebreak, target IP addresses are simply unreachable from ISP customer networks and endhosts. Rather, IP packets are addressed to proxies deployed near the edge using IP anycast, and from there are tunneled using the target IP addresses. This use of IP indirection, as well as the use of IP anycast and tunneling to deploy firebreak, is the main research contribution of firebreak. This paper describes the firebreak architecture, discusses its pros and cons, and suggests directions for future work.

## 1 Introduction

In spite of the fact that DDoS is one of the most serious problems facing the Internet today, no effective and viable solutions exist. Practically speaking there are two basic approaches in effect today. One is simply to have more bandwidth and server capacity than the attacker. We understand that Microsoft, to a large extent, protects itself this way [1]. This approach is perhaps, at least for the time being, adequate for very rich companies, but is not feasible for small companies or individuals. In any event, this is clearly not a good long-term strategy even for rich companies.

The second practical approach is offered by CDNs like Akamai. Here, DNS is used to direct HTTP queries to proxies richly deployed around the Internet. By their sheer scale, these proxies can absorb a DDoS attack (or, almost equivalently, a surge in legitimate traffic). This approach requires that the origin server IP address be kept secret. It also requires that the DNS infrastructure be large enough that it itself cannot be attacked. Indeed, this DNS was apparently the target of a recent attack on Akamai (June 15, 2004). The attack was reported to have a small but noticeable impact on Akamai's service.

Alternatively, if an attacker somehow learns the IP address of the origin server of the web service it wishes to attack, then it could bypass Akamai's proxy defenses altogether, especially for active content, which is difficult to serve out of web proxies. This amounts to security through obscurity, which is widely considered to be a bad security policy.

On the research side, two broad approaches have been identified. One is traceback/pushback, whereby routers cooperate to discover the sources of an attack (traceback) [2][3][4] and install filters to remove the bad traffic as near to the sources as possible (pushback) [5][6]. [7] suggests that the main weakness of these approaches is that they cannot adequately distinguish the good traffic from the bad.[1] While there is some truth to this, we believe that there is still value in such filtering, in part because they offload the targets so that the targets can still reasonably service traffic (good and bad), but also because the filters will still remove more bad traffic than good, primarily because of their proximity to the bad sources.

Our objection to the traceback/pushback approaches, as well as to [7], is much more pragmatic: they require too tight a coupling with the routing fabric. The traceback/pushback functionality, whether implemented in routers or boxes sitting next to routers, needs to exist in multiple points on the path between attacker and target to be effective. This in turn requires close cooperation between ISPs (and therefore

[1] [7] also has a nice and succinct description of the referenced traceback, pushback, and overlay approaches which we don't repeat here.

agreement between ISPs on what protocols to use, among other things). Such approaches require a lot of economic lift to get off the ground.

The other broad research approach is to deploy some kind of overlay [8][9][12]. These approaches overcome the deployment problems associated with the more router-oriented approaches, but have other shortcomings. [7] argues that the authentication secrets [8] and[9] use are shared among too many components, and are therefore subject to compromise. Once again, our objections are more pragmatic. SOS [8], Mayday [9], and Si3 [12] all require users to traverse an overlay that provides some form of protection. But these approaches require either that the target IP address remain secret, or that there is router-based protection in the vicinity of the target to defend against attackers. After all, attackers have no motivation to try to reach the target via the overlay (or via web proxies, as with the CDN approach). They can bypass the overlay (proxies) by sending packets directly to the IP address of the target. The argument that SOS and Mayday make is that such router filtering can be lightweight and therefore deployed in high-speed core routers, but we remain somewhat skeptical of this argument. Routers are highly constrained to do a minimum amount of processing, so even relatively simple filtering is a hard sell. In addition, these schemes require changes to legitimate endhosts to use the overlays. [12] suggests an alternative using IP-level indirection to avoid changes to endhosts, and is similar to firebreak in this regard. However, the approach they outline requires per-flow state and NAT operations in existing edge routers.

## 2 Firebreak Architecture

As should be apparent from the above discussion, one of the primary goals of Firebreak is to design a DDoS defense that is deployable in both a technical *and* economic sense. Towards this end, we want a solution that requires no changes to endhosts or current router software. We want a solution for which a viable business model exists and can support whatever initial infrastructure investments are necessary. We also recognize that DDoS is an arms race─we don't expect a silver bullet. Rather we expect to make the cost of defending against reasonable attacks as small as possible (and certainly much smaller than the damage caused by such an attack).

Beyond this, we take it as a given that the point of defense, that is the boxes where attack packets are filtered, should be as near the source as possible. There are two reasons for this. First, the further away the defense is from the target, the more bandwidth there is to absorb the attack. Second, the closer each box is to the source, the fewer sources each box has to filter against. Towards this end, the solution should be deployable across multiple ISPs. This in turns means that minimal or no coordination be required between ISPs deploying the solution.

The key concept behind firebreak is the use of IP-level indirection. The basic idea is that source end hosts (friend and foe alike) simply cannot send IP packets directly to target hosts (e.g. in Figure 1, packets from host S addressed to T1). IP reachability simply does not exist─IP packets addressed to the target are dropped by legacy routers very near the source. Instead, packets must be addressed to intermediate boxes, called *firebreak boxes* (or more typically just *firebreaks*) deployed near the edge. The firebreaks map these *firebreak addresses* into the *target addresses*, and tunnel the packets to the target. For instance, in Figure 1, S addresses packets to F1[2], which gets delivered to a nearby firebreak (FS). The firebreak maps F1 into the target address T1, and tunnels the packet to the target using its own unicast address FSu as the source address of the outer header. If the target itself is incapable of de-tunneling these packets, a proxy placed in the physical path near the target can do it (or, if necessary, the firebreak could NAT the packet). Reverse packets actually take a different path back, as explained in Section 1.2.

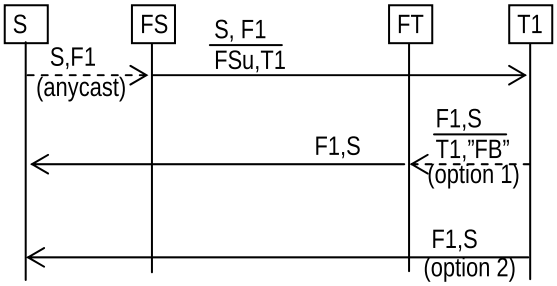

**Figure 1: Packets between non-protected host S and (protected) target T1. Option 1 is where T1 cannot spoof its source address, option 2 is where it can.**

DDoS *monitor* functions exists at the target (though firebreaks may feed statistics to the monitors to aid them). Normally firebreaks simply pass all traffic to the target. When a monitor detects an attack or an overload, however, it sends control messages to the appropriate firebreaks (e.g. to address Fu) requesting the appropriate defensive action (later we discuss what this may be). It may turn out for instance that some firebreaks do not have any or many attackers behind them, and therefore don't need to be told anything.

[2] Host S may have obtained address F1 through DNS, for instance.

Of course, we don't expect to see firebreaks immediately deployed at every edge router in the Internet, so we need a strategy for incremental deployment of firebreaks (accepting that the initial deployment in any event must be big enough and disperse enough to absorb a sizable attack). For this, IP anycast is proposed [10]. All firebreaks advertise the complete set of firebreak addresses into the routing fabric. As a result, packets addressed to any firebreak address are routed to a nearby firebreak. If a firebreak fails or is taken out of service, IP routing will failover to another, operational firebreak.

In fighting forest fires, a firebreak is a long swath of cleared vegetation that contains the fire within some region of the forest, as far as possible from protected resources like homes. We view our network firebreak as a long swath of protection that contains DDoS attacks near the sources and far from the targets. The firebreak can also be viewed as a "reverse firewall". A firewall is deployed near the protected target, and is dedicated to protecting only that target. An individual firebreak, by contrast, is deployed near a set of sources, protects all targets (partially), and each target controls that aspect of the firebreak that pertains to itself.

Though there are many details yet to examine, let's see how this basic architecture addresses the problems outlined so far. The target IP addresses do not need to be kept secret, because the edge routing infrastructure won't allow packets to reach them. No router changes are required, either near the target or throughout the routing fabric. As such, no special coordination is required between ISPs. Indeed the firebreak can be deployed along the same economic model as a CDN, though extensive coordination is required between the CDN and the ISPs where it deploys. No changes are required to non-protected hosts, because IP anycast is used to direct client packets to nearby firebreaks. Protected hosts must either be modified or front-ended by in-the-path proxies, but we believe it is reasonable to assume that firebreak customers would be willing to do this.

# 3 Design Details

## 1.1 Route manipulation

At the core of firebreak is making targets unreachable from endhosts but not firebreaks. Indeed, this aspect of firebreak is applicable to any overlay-based DDoS scheme, including Akamai's, as it eliminates the need for keeping target addresses secret and for having router-level protection near the target.

Because firebreaks would not be deployed in all ISPs, lets refer to ISPs that have deployed firebreaks as Firebreak ISPs (F-ISP) and ISPs that have no deployed firebreaks as Non-firebreak ISPs (NF-ISP). Preventing packets from NF-ISPs from reaching targets is straight-forward: simply refrain from advertising those prefixes from F-ISPs to NF-ISPs. Alternatively, the F-ISP could advertise the target prefixes, but immediately forward those packets into a black hole at the peer. The disadvantage of the former approach is that the NF-ISP could potentially on its own forward target packets to the F-ISP─the latter approach avoids this.

Route manipulation within an F-ISP is more involved, because some hosts (firebreaks) must have IP reachability to targets, while other hosts (users) must not. Figure 2 shows a simplified but for our purposes accurate topology of a single POP (Point of Presence) within an ISP. ISPs typically consist of multiple such POPs. A POP usually contains at least two types of routers: Access Routers (AR) are used to aggregate customer networks and feed into a smaller number of higher capacity Core Routers (CR). Core Routers have long distance links to other POPs within the same ISP, and to other ISPs.

Figure 2 also shows three valid deployment options for firebreaks (labeled FB). In the first, the firebreaks are attached to access routers, but via an interface that is distinct from the interfaces used to aggregate customers. In this deployment, the filtering rules in the access router are configured such that packets received via a customer interface and destined to a target prefix are dropped. Off-the-shelf routers have these filtering capabilities, but the problem is that turning them on slows down router performance. Especially, there may be many target prefixes, so a large number of filtering rules might be required in the access routers.

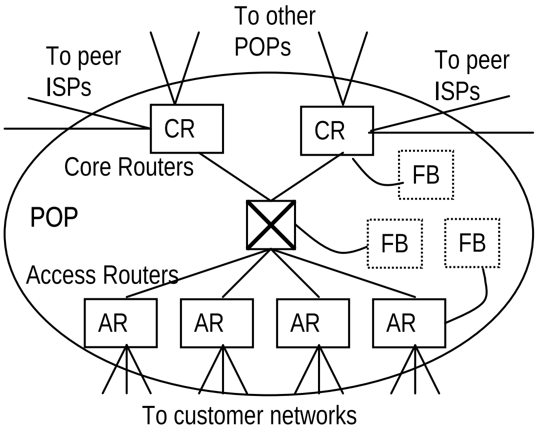

**Figure 2: An ISP POP with three firebreak deployment options**

In the second option, the firebreaks are attached to core routers. The disadvantage to this deployment is that the firebreak is now further away from the source (customer), and so has to discriminate among a larger number of sources (i.e. all those feeding into the POP, rather than those feeding into one or a few access routers). The advantage of this deployment, however, is that forwarding rules, rather than filtering rules, can be used. Specifically, BGP is manipulated so that the access routers have no forwarding table entries for the target prefixes at all. Or, if the access routers use default routes to the core routers, they are given explicit forwarding table entries for the target prefixes, but the packets are forwarded to a black hole. Either way, since the normal forwarding function is used to prevent access to targets, the access routers operate at normal full speed.

In the third option, forwarding rules are used as described in the previous paragraph, but rather than attach the firebreaks to the core routers, they are attached at the link layer to high speed switches used to interconnect the various routers within the POP. The advantage here is that the firebreaks can see which access router handled the packet, and so have better source discrimination. Multiple firebreaks may be deployed at a single switch through any number of techniques (manipulating "ARP" tables in the access routers, through VLANs, or with an L3 load balancer).

**Deployment with IPv6**

While this paper is primarily written with IPv4 in mind, it is worth mentioning that there is a mode of operation with IPv6 (non standard) that would greatly simplify the firebreak. Specifically, the IPv6 address space could be split in two, half for targets and half for firebreaks, with a simple mapping between them (i.e. the flipping of a single bit). This mapping would be defined as a standard, and coded into the fast path of access routers. In this model, the network administrator would only need to identify the customer interfaces on the access router, which would then do the proper filtering.

### *1.2 Packet formats*

Now we give a more complete description of how packets are forwarded through firebreaks. We have several requirements for our approach. First, the recipient of a packet must know to send a return packet from inspecting only the contents of the received packet. Clearly legacy hosts must be able to do so in the standard way. Second, a protected target must be able to initiate connections and still remain protected. Third, the recipient of a packet, if a protected target, must learn the unicast address of the traversed firebreak, so that it may send control messages if necessary. Finally, where possible the addresses of the inner IP header must not change E2E. The exception to this is where the protected host is a legacy host, and an in-the-path proxy translates the headers. The remaining description assumes no such proxy.

To accomplish these goals, protected hosts always transmits packets with their firebreak address as the source address. If the protected host is not able to spoof its source address (for instance because an edge router drops source spoofed packets), then it must tunnel its packets to a nearby firebreak (option 1 in both Figure 1 and Figure 3). It can do this by transmitting the tunneled packet to a generic firebreak address (shown as "FB").

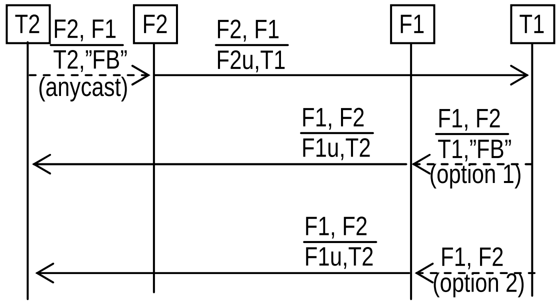

**Figure 3: Packets between two protected hosts. Option 1 is where T1 cannot send source spoofed packets. Option 2 is where it can. (The same applies to T2.)**

When a firebreak receives a packet from a protected host for a non-protected host, it simply strips the outer header and forwards the packet (Figure 1). When a firebreak transmits a packet to a protected host, the inner header is the same as that received. The destination IP address in the outer header is that of the target host, as mapped from the received firebreak address. The source IP address in the outer header is the unique unicast address of the firebreak. This is stored by the target host (or a monitor acting on its behalf) to later send control messages to the firebreak. Note that firebreaks can and presumably would be themselves protected by the firebreak, so the IP address they use is actually their unique anycast firebreak address.

Looking at Figure 1 and Figure 3, it should be clear that either end could have initiated the packet exchange with the headers shown. Note also that a protected host, as an initiator, does not need to know if the destination is a protected host or not. The firebreak must know, but firebreaks are required to know every mapping in any event.

Finally, we note that this style of encapsulation requires that all of a protected host's traffic go through nearby firebreaks – a potential bottleneck. For communications with non-protected hosts (Figure 1), this can be avoided in cases where the target can transmit packets with spoofed source addresses, which in many but perhaps not all cases could be arranged.

### 1.3 Firebreak Control

As discussed previously, filtering rules in firebreaks are controlled by monitors at the target. Specifically, a given target T1 has the authority to install firebreak rules that relate to itself, but does not have the authority to install firebreak rules about any other target. Therefore, targets must be authenticated by firebreaks. This could be done by having the protected host establish a secure connection with its nearest firebreak (e.g., FT in figure 1), through which it forwards control messages. Firebreaks could in turn authenticate control messages between each other.

### 1.4 Firebreak defense rules

As already stated, we expect firebreak to evolve with increasingly sophisticated attackers. Even so, a few simple firebreak defense rules, outlined here, could go a long way.

Note that a firebreak can over time build up knowledge of what source addresses it legitimately expects to see by monitoring successful TCP connections. If an attack uses randomly chosen spoofed source addresses, the firebreak could be told to filter all packets that appear to be spoofed. A counter attack to this defense would be for the attacker to spoof only addresses near its own address. In this case, the firebreak could be told to start locally completing TCP handshakes.

If the attacker does not spoof source addresses, and appears to send legitimate traffic, the firebreak could be told to fair-queue based on source IP, so that legitimate users can get through. If the attack is so large that even with fair queuing legitimate users don't get through, then higher-level authentication mechanisms may be used. For instance, for web applications, the firebreak could be told to start transmitting web pages asking the human user to type in a human-readable but not machine-readable code.

The point here is not to present an exhaustive list of defense mechanisms, or even to pretend that we understand what such an exhaustive list would look like, but rather to illustrate that a firebreak could utilize a rich and increasingly sophisticated set of defenses.

## 4 Firebreak Issues

In this section, we discuss a number of possible objections to the firebreak architecture.

**Doubles the number of addresses required:** This is basically true, and must be accepted as one of the costs of DDoS protection. Having said that, we note that it is possible for multiple targets to share the same firebreak address if the firebreaks also use the port number to map firebreak addresses to target addresses.

**Scales poorly by the number of target addresses:** Since both edge routers and firebreaks maintain per target address prefix information, it is important to insure that target addresses are allocated from large blocks of addresses dedicated to the targets. For instance, each ISP that hosts protected targets (i.e. in their data centers) could take target addresses from a block dedicated to target addresses. Likewise firebreak addresses should be allocated from corresponding blocks, so that the firebreaks only essentially have to map prefix block to prefix block.

**Scales poorly by the number of firebreaks:** One objection might be that there are too many firebreaks for any given controller to manage in a short amount of time. Practically speaking, we doubt that this is an issue. A controller should reasonably be able to install filter rules in several hundred firebreaks per second.

**Firebreaks introduce another point of failure:** It is true that if a firebreak goes down, all of the flows traversing it will fail until routing can converge to a new firebreak. However, we would expect to see multiple firebreaks deployed per POP, so the loss of a firebreak would almost always require only very local re-routing, which can be done quite fast. In other words, this problem can be mitigated by good engineering.

**The firebreak itself can be attacked:** Obviously we need to insure that the firebreak itself doesn't introduce new vulnerabilities (in the same sense that Akamai's use of DNS creates a vulnerability). Assuming that monitors are indeed authenticated, and that the firebreaks themselves are protected hosts, and of course that there are enough firebreaks to absorb a brute force attack, we don't see an obvious way to attack the firebreak. One approach might be to try to spread attacks over many targets, attacking each just enough to get it to trigger defenses in firebreaks. The defense mechanisms themselves would slow down the firebreaks, and might also cause some legitimate traffic to be blocked through imperfect filtering. While we clearly need to carefully engineer for this and other possible attacks, we can at least say that we've raised the bar for the attackers.

**Initial firebreak deployment must be large:** Even if there is only one protected target, the initial firebreak deployment must be large enough to repel the largest expected attack. This could easily amount to hundreds of firebreaks over 10 or 20 ISPs. So while mechanistically the firebreak is incrementally deployable, in practice it is not. Having said that, we believe our approach minimizes the amount of required initial investment compared to approaches that require changes to routers or client hosts.

**Large-scale anycast is not well understood:** We have some concerns that large-scale anycast may have some bad unintended interactions with global BGP. For instance, having a large number of advertising the same prefixes could result in a large number of BGP events, which could in turn result in hold downs. Given that BGP dynamics themselves are poorly understood and constantly evolving, this is an area that requires extensive experimentation.

# 5 Status and conclusions

The status of our work at this time is as follows. We are working to deploy a general IP anycast infrastructure (called PIAS, Proxy IP Anycast Service). The software is implemented and working in the lab, and we have preliminary approval for obtaining the required IP address blocks. We expect to start a small deployment soon. From this, we will be able to experiment with IP anycast on a larger scale than previously achieved. In the meantime, we are starting to talk to ISPs in order to extend PIAS to include firebreak functionality.

We are also interested in a number of extensions to firebreak. For instance, as described in this document, firebreak is appropriate for the "public" client/server model of communications. In this model, firebreaks by default take no defensive action unless otherwise instructed by targets. A much more challenging problem would be that of protecting "private" peer hosts. In this model, firebreak behavior is the opposite: don't allow any packets unless otherwise told by the target. This in turn means that standing filter state always exists in firebreaks. This, as well as the fact that there are potentially many more peers the servers, presents significant scaling difficulties.

We are also interested in the use of firebreaks to control worms. Because firebreaks are deployed throughout the Internet, they are in a good position to help detect worms or other port scanning activity.

Another thing to consider is whether the firebreak model extends to intranets. The main problem here is that intranets increasingly provide much or most of their connectivity at layer 2, so it is not clear if the firebreak model really applies.

**Conclusions**: This paper presents the firebreak architecture, which uses IP-level indirection to decouple senders from protected receivers at the IP level. Through the use of IP anycast, routing policies, and tunneling, we describe how the firebreak can be deployed as a proxy infrastructure with no changes to routers or sending endhosts, and with minimal changes to routing policy configuration. We believe that the firebreak is the most general and economically viable approach to DDoS proposed so far.

## References

[1] Private communications.

[2] S. Bellovin. ICMP Traceback Messages. http://www.research.att.com/~smb/papers/draft-bellovin-itrace-00.txt, Internet Draft), 2000.

[3] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. "Practical Network Support for IP Traceback,". In Proc. ACM SIGCOMM 2000

[4] A. Snoeren, C. Partridge, L. Sanchez, C. Jones, F. Tchakountio, S. Kent, and W. Strayer. "Hash-Based IP Traceback," In Proc. ACM SIGCOMM 2001

[5] R. Mahajan, S. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker. "Controlling High Bandwidth Aggregates in the Network," Computer Communications Review, 32(3), July 2002

[6] J. Ioannidis and S. Bellovin, "Implementing Pushback: Router-Based Defense Against DoS Attacks," In Network and Distributed System Security Symposium, 2002

[7] Tom Anderson, Timothy Roscoe, David Wetherall, "Preventing Internet Denial-of-Service with Capabilities," ACM Hotnets II, Nov. 2003

[8] A. Keromytis, V. Misra, and D. Rubenstein, "SOS: Secure Overlay Services," In Proc. ACM SIGCOMM 2002

[9] D. Andersen, "Mayday: Distributed Filtering for Internet Services," In Proc. of USITS 2003.

[10] C. Partridge, T. Mendez, W. Milliken, "Host Anycasting Service", RFC 1546, November 1993

[11] K. Gummadi, S. Saroiu, S. Gribble, "King: estimating latency between arbitrary internet end hosts," ACM Sigcomm IMW 2002

[12] Karthik Lakshminarayanan, Daniel Adkins, Adrian Perrig, and Ion Stoica, "Taming IP Packet Flooding Attacks," ACM SIGCOMM HotNets-II, Cambridge, MA, November 2003