Cookies Along Trust-Boundaries (CAT): Accurate and Deployable Flood Protection

Martin Casado Aditya Akella Pei Cao Niels Provos Scott Shenker
{casado,cao}@cs.stanford.edu,aditya@cs.cmu.edu
niels@google.com, shenker@icsi.berkeley.edu

Abstract

Packet floods targeting a victim's incoming bandwidth are notoriously difficult to defend against. While a number of solutions have been proposed, such as network capabilities, third-party traffic scrubbing, and overlay-based protection, most suffer from drawbacks that limit their applicability in practice.

We propose CAT, a new network-based flood protection scheme. In CAT, all flows must perform a three-way handshake with an in-network element to obtain permission to send data. The three-way handshake dissuades source spoofing and establishes a unique handle for the flow, which can then be used for revocation by the receiver. CAT offers the protection qualities of network capabilities, and yet does not require major architectural changes.

1 Background and Motivation

Denial-of-service (DOS) via packet flooding remains a serious problem on the Internet today. Receiver centric solutions are generally ineffective against flooding because the resource being exhausted is not under the victim's control. Other forms of resource consumption DOS, such as outgoing bandwidth exhaustion or computational complexity attacks, can be treated as a local resource management issue and are often dealt with at the victim with resource scheduling [21] or admission control [15] schemes.

Industry and the Internet research community have proposed or deployed a number of solutions which have made great strides towards offering protection from flooding. We describe some of them here:

Traffic filtering near the victim: ISPs use a variety of statistical and machine learning techniques to "infer" attack traffic with little input from victims, e.g. Push-Back [16], and Arbor Networks [1]. The location where the filtering is performed is either at the tier-1 ISP [1], pushed to ISPs close to the sources [16], or pushed to points upstream of the local bottleneck link of the victim [13]. In general, these schemes provide a trade-off between the rate of false positives and the degree of sensitivity to low-bandwidth attacks.

Traffic filtering near the source: Other schemes suggest filtering packet floods as close to the source as possible, e.g. AITF [7] and Firebreak [11]. However, we believe such schemes are unlikely to be adopted because they involve asking an ISP to take damaging action against one of their customers on behalf of a victim that they have no economic or trust relationship with[1].

Overlays: Overlay solutions, such as CDNs, Mayday [6] and SOS [14], filter packet floods at a large set of nodes distributed across the edges of the network. The aggregate bandwidth offered by the overlay nodes can be very high. However, even in the presence of a high-bandwidth overlay to vet traffic, the back-end server's public IP address remains vulnerable.

Network Capabilities: In a departure from filtering-based approaches, network capability schemes, such as SIFF [22] and TVA [23], require recipients to mark wanted/legitimate traffic with unforgeable capabilities. These schemes guarantee that legitimate traffic always receives priority over illegitimate traffic. However, existing proposals require changes to both clients and servers, as well as upgrades to routers to honor capabilities and protect legitimate traffic. As a result, they do not provide tangible near-term benefits.

Third party traffic scrubbing: In this approach, a company peers with a number of large ISPs, and sinks all traffic to the protected servers. The company can then offer DoS prevention services to the web-servers such as protection from connection floods (by performing the TCP handshake in the network), flood detection (using statistical techniques mentioned above) and per-flow rate limiting (e.g. Prolexic [3]). However, with this approach it is difficult for web sites to control decisions regarding the legitimacy of traffic. Generally, the company offering protection handles both the detection of malicious traffic

[1] [13] provides a good argument in support of this position

and the enforcement.

In this paper, we propose CAT, a new approach for protecting public web servers against bandwidth floods. Our goal for CAT is to offer the same protection guarantees as capabilities while having a practical adoption story. This implies no major architectural changes, backwards compatibility with existing clients and an incremental deployment path. CAT achieves this goal through combinations of the following techniques:

- *flow cookies*: This is a stateless, backward-compatible capability mechanism that offloads connection-setup and filtering to in-network elements without requiring per-flow state.

- *filtering on trust boundaries*: Existing commercial relationships among ISPs can be leveraged to identify viable locations for filtering. We term the furthest set of ISPs with which a web server has an economic relationship its *trust boundary* and argue that the trust boundary is a natural location for filtering.

All flows to a CAT protected server must first perform a handshake with a middlebox on the server's trust boundary. Successful completion of the handshake will provide the sender with a valid flow cookie which will allow the sender to communicate with the receiver for the lifetime of the cookie or until the receiver revokes it. Cookie revocations are performed by the middlebox on the trust boundary.

In the following sections we present an overview of CAT (§2), then discuss flow cookies in more detail (§3) and a mechanism for dynamically detecting the trust boundary (§4). In §5 we present an analysis of BGP data to determine the properties of existing trust boundaries. Finally, we present related work in §6 and conclude in §7.

2 Overview of CAT

CAT was designed around two guiding principles:

1. For filtering mechanisms to be accurate and effective, decisions about whether or not to permit flows must be made by devices or services that can reliably determine both the *origin* of traffic as well as whether the destination *wishes to receive* the traffic.

2. To ensure incremental deployability of a protection mechanism, existing commercial relationships between network entities (e.g. ISPs) can be leveraged as forcing functions for the deployment of protection infrastructure.

The *flow cookie* mechanism realizes the first principle. Flow cookies liberally borrows ideas from capabilities and third-party traffic scrubbing. Like third-party scrubbing, a "cookie box" is situated on-route between the clients and protected web-server at a network location with high incoming bandwidth (typically inside the protected server's ISP). The cookie box engages all flows from clients in a three-way handshake. Only connections that have completed the handshake are forwarded on to the web server. Subsequent to the handshake, flow cookies functions similar to network capabilities. The cookie box inserts capabilities in outgoing packets from the server, and these capabilities must be echoed in future packets from clients. We use the TCP timestamp field to insert the capability, thereby ensuring backwards-compatibility. When the web-server deems a client to be misbehaving, it can terminate the connection and/or simply request the cookie box to filter the offending flow or IP.

This simple approach offers several salient features:

- Offloading the three-way handshake to high-bandwidth infrastructure allows special purpose hardware to determine if clients are source spoofing.

- All packets forwarded to the server must carry a non-forgeable cookie which is computed over the packet header. Therefore, web-servers can make effective policy decisions on the basis of IP addresses which are enforced in the network.

- The signaling of the "wanted-ness" of a flow is implicit via response traffic from the protected server. That is, if a server does not reply to a client with an ACK, the client will be unable to get a fresh cookie after its current one times out.

- With filtering enforced in-network, this approach offers web servers the protection bandwidth of the link(s) over which the cookie box is deployed.

- Detection of illegitimate traffic is performed at the web server, hence the server can implement flexible and meaningful policies for its particular service.

As described above, flow cookies operates through a point deployment, and is therefore easy to adopt. However it suffers from two key limitations: (1) if an attacker overwhelms the link on which the cookie box is deployed, she can effectively flood the server (2) unless the cookie box sinks all traffic, packet floods can avoid filtering by routing around the cookie box (e.g. using alternate ISPs or peering points).

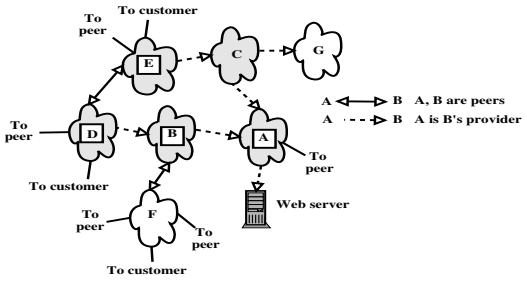

Figure 1: ISPs shaded gray are in the region of trust of the web server. ISPs enclosed in a white square are on the trust boundary For example, ISPs A and B are on the trust boundary since they terminate the trust chain for traffic received from their peers.

Naturally, these limitations can be addressed by deploying more cookie boxes at a diverse collection of network links spread across several ISPs. We note that deployment of cookie boxes can be facilitated by the web server taking advantage of the network links for which it, or its ISP, or recursively its ISP's ISPs, have an *economic relationship* (Two network entities have an economic relationship if one *pays* the other for delivering traffic). The transitive closure of such economic relationships among ISPs defines a "region of trust" for each protected server (see Figure 1). We consider ISPs in the trust region that terminate a sequence of economic relationships to be on the *trust boundary* (Figure 1).

Links on a server's the trust boundary are natural locations for deploying cookie boxes because they offer the highest incoming bandwidth without requiring the web-server to forge new relationships. We argue that not only does a server have stronger clout to persuade an ISP within its region of trust to deploy a cookie box, it also has greater leverage to ensure that the cookie boxes will be administered and operated correctly

To be effective, a trust boundary must have the following two properties. First, the aggregate bandwidth across all boundary links must be very high. This ensures that cookie boxes on the trust boundary can handle high-volume flooding. Second, the number of routes traversing the boundary links should be roughly proportional to the bandwidth of the links. This prevents in-network hot-spots from underprovisioned links managing too many flows while other links on the boundary have available filtering capacity. In §7, we present an analysis of AS level connectivity graphs for the Internet which demonstrates that the trust boundary for most stub networks today satisfy both of these properties.

By definition, ISPs on the trust boundary of a given web server must perform the TCP handshakes and subsequent filtering on all traffic to the web server. To effectively provide this functionality in a setting where ISPs could lie on one of several trust boundaries (and inside several other regions of trust), the ISPs must coordinate among each other to decide who must perform the TCP handshake and subsequent filtering for a particular web server. In §4 we present a simple modification to BGP by which multiple ISPs can ascertain their responsibility in a distributed manner.

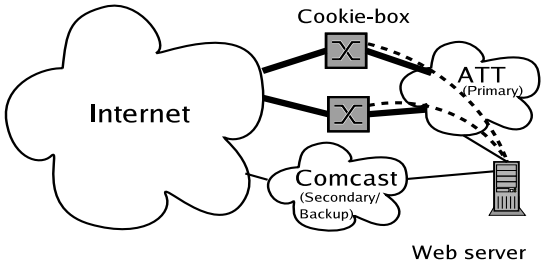

Figure 2: A multihomed web server seeking protection deploys cookie boxes at some of the network links of its primary ISP. These could be edges along which the web server expects to receive a large fraction of its traffic.

3 Flow Cookies

Flow cookies is an extension of SYN cookies[8] wherein a middlebox places a secure, limited lifetime cookie within the TCP timestamp of every outgoing data packet from the protected server. Flow cookies offers strong protection against flooding, does not require modification to clients or to the network, is resistant to source-spoofing, and does not require per-flow state in the network.

Unlike approaches common in the commercial world, flow cookies is not meant to be deployed at a web server's perimeter. Instead, a "cookie box" (or a handful of boxes) which enforces flow cookies are deployed at the edges of the protected web server's ISP. We illustrate an idealized setup in Figure 2.

In this section, we present a brief overview of the flow cookies approach. Flow cookies was designed to protect services using TCP. We assume that other protocols, such as UDP, are managed through other means, such as rate limiting aggregates. The full details of the flow cookie protocol, design issues and analysis may be found in [9].

3.1 Overview

The cookie box(es) and the protected web server cooperate to perform the following four tasks:

1. The cookie box intercepts all SYN packets destined to the web server. If the SYN packet's source IP is in an "`IP blacklist`", it is dropped (i.e., the IP blacklist is only looked up for SYN packets). Otherwise, the box responds with a SYN cookie [8] in which the source address is forged to be that of the web server. The cookie is computed using a keyed message authentication code

over the connection 4-tuple. SYN cookies does not require state maintenance at the cookie box and can be run at gigabit line speeds [2].

This first step ensures that spoofed sources, SYN floods and unwanted connection attempts cannot traverse the link between the cookie box and the protected website.

2. For packets to the web server that carry an ACK flag (this includes all data packets), the cookie box checks that the ACK'ed sequence number is a valid SYN cookie. If it is, the connection is handed off to the web site using a TCP handoff scheme such as [4].

3. For outgoing packets from the web server to its clients (this happens after the web server has accepted the client connection request), the cookie box adds a secure non-forgeable "flow cookie" (explained below).[2] The flow cookie is computed in a similar manner as the SYN cookie and is valid for a limited amount of time (e.g. 1 minute). As we explain below, the flow cookie is echoed back by the client, and checked by the cookie box.

The cookie box implicitly infers whether the web server wishes to engage in communication with a particular end-point by marking outgoing ACKs. If the web server chooses not to send ACKs to a client, the client will be unable to get fresh cookies once the old one times out. The flow cookie further helps the cookie box ensure that only packets belonging to flows accepted by the server traverse the link between the cookie box and the web server.

4. The web server understands the accepted usage policy of its local administration, and is already keeping per-flow state for each outstanding connection. Therefore, it is in the best position to determine if a client is misbehaving.

If the web server does not want to receive packets from a particular flow, it can do two things: (1) push filters to the cookie box's "`flow blacklist`"; in this case, the cookie box maintains the source IP and port in a revocation list with an associated time out and filters packets accordingly. Or, (2) inform the cookie box to stop issuing fresh capabilities for the client. This can be done statelessly by simply closing the connection, in which case the client will no longer receive valid cookies for the flow (after the current cookie times out). The first approach can be employed to filter high-bandwidth malicious flows immediately. The second approach can be used in less critical situations or to shut off low priority clients when under overload.

If a server determines that an attacker is behind a given IP address (or address block), it can request the cookie box to add the IP (address block) to the `IP blacklist`.

[2]We assume that Internet routes are symmetric at the AS-level.

3.2 Ensuring Backwards-Compatibility

To be backwards-compatible we exploit the *TCP timestamp* option and place the flow cookie from step #3 in the timestamp field of packets.

The TCP timestamp option, proposed in RFC-1323 to measure RTTs, is supported by all the major host operating systems. According to the RFC, once the option is enabled by both ends, the sender places a timestamp in a packet, and subsequent packets from the receiver *echo* the timestamp. Common operating systems enable timestamps by default, with the exceptions of Windows2000 and WindowsXP. As has been shown by [20], it is possible to trick Windows into echoing timestamps by including a timestamp option in the SYN ACK packet.

On connection set up, the cookie box negotiates the timestamp option with the client. RFC-1323 does not specify how the timestamp value is to be encoded in the option field. To prevent disruption, the cookie box and the web server must explicitly agree on a format. Also, care needs to be taken if the web site wishes to use timestamps to measure RTTs. To avoid undesirable interactions, protected web servers could be dissuaded from using timestamp values in RTT calculations. This requires a relatively insignificant modification to the TCP stack at the web server. Also, all network stacks are able to do RTT calculations without the aid of timestamps. An alternate method that does not require any modification of the web server is described in full detail in [9].

3.3 Implementation

To demonstrate flow cookies' ability to inter-operate with existing clients, we implemented flow cookies within a software router using a user-space network development environment [10]. We tested our implementation using numerous public web servers. Our tests show that flow cookies is compatible against various client OSes and server software and can handle fully dynamic, interactive sessions as well as static content.

We also found that the flow cookie implementation and IP blacklist lookup had little effect on the throughput of our router and was able to operate at gigabit speeds during micro-benchmarking. While our focus was on a software implementation, we believe that flow cookies can be easily implemented in hardware.

We also tested against synthetic flooding attacks and found that flow cookies is indeed able to offer full protection of established flows even under severe loads. Full details of our implementation and testing are available in [9].

4 Leveraging Trust Relationships

A primary limitation of flow cookies as described thus far is that it only leverages the protection bandwidth of a few high-speed links. So either the protected links must sink all traffic to the web server (this may require changes to routes and could introduce network bottlenecks) or the server is vulnerable to attacks on unprotected links. A more troublesome problem is that, when several cookie-boxes are deployed across many ISP—where each cookie box could be protecting a distinct set of web servers—the cookie boxes must arrive at a consensus on which of them should perform the handshake and filtering for a particular flow.

In this section we extend flow cookies from being a point-solution to a more generic wide-area service. We introduce a minor modification to BGP that allows ISPs which have deployed cookies-boxes to determine which boxes manage the handshake and do the filtering on a per destination basis. We also argue that the most viable and effective deployment strategy is to leverage existing client/provider relationships.

4.1 Trust Relationships Between ISPs

In the Internet today, neighboring ISPs rely on contractual agreements or SLAs to coordinate mutual exchange of traffic. While some agreements involve money changing hands (such as customer-provider contracts), others are more faith-based (e.g. peering relationships) [17].

We argue next that such agreements also define "trust relationships" between ISPs, either implicitly or explicitly. By a trust relationship between network A and network B, we mean that A trusts B to take the necessary steps from preventing packet floods from entering into A's network via the A–B link.

In the simplest case, if ISP A pays ISP B for global Internet connectivity, then it is safe to say that "A implicitly trusts B"; a violation of this trust – detectable at A, for example, when B lets packet floods through – will likely result in A picking an alternate provider. Such implicit trust relationships already exist today.

Neighboring ISPs, that don't share a customer-provider relationship, may also enter into *explicit* trust relationships. For example, two peer ISPs may contractually agree to mutually cooperate and vet packets destined for each other's network. Several such instances of cooperation among peer ISPs for traffic engineering purposes have been observed empirically. Explicit trust relationships may also be forged between a customer network and a non-neighbor ISP providing a specialized form of protection.

By default, trust relationships are not symmetric. For example, a customer may trust its provider to do the right thing, but the provider cannot assume any guarantees from its customer. However, most explicit trust agreements between peers are likely to be symmetric.

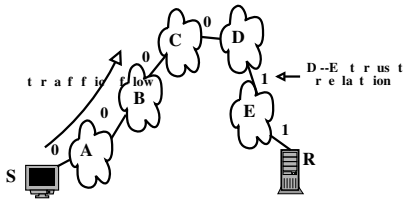

Figure 3: An illustration of trust relationships on an end-to-end AS-level path.

In what follows, we assume trust exists in the customer-provider relationships, not in peering relationships.

4.2 A BGP-based Mechanism

Our insight is that we can leverage BGP and existing trust relationships between ISPs to determine, in a completely distributed and stateless manner, a single ISP that is responsible for inserting flow cookies and filtering packets.

We illustrate our solution through an example. Say source S wants to send to receiver R. By the design of routing in the Internet, S knows the "AS-level path vector" to communicate with R via BGP: `A, B, C, D, E` (Figure 3). Here A is S's ISP and E is R's ISP.

We assume that every ISP along the path vector knows the relationships between pairs of ISPs downstream from S towards R. For example, A knows the trust relationship for `A--B`, `B--C`, `C--D`, `D--E`, and `E--R`. B knows `B--C`, `C--D`, `D--E`, and `E--R` and so on.

This information is easy to piggy back on to the BGP path vector protocol: In BGP, when an AS wants to announce a route to a neighbor, the AS appends its AS number to the announcement. We simply change the announcement to also indicate whether the AS trusts the neighbor. The destination prefixes to be protected are similarly advertised in the BGP announcement.

Lets says that the trust relationship between a pair of neighboring ISPs, e.g., A-B, is encoded in a binary form: 1 means that B trusts that packets received from A have been vetted by A using flow cookies, [3] and 0 means otherwise.

Assume the trust relations on the path from S to R are as follows:

```
S--A: 0
A--B: 0
B--C: 0
C--D: 0
D--E: 1
E--R: 1
```

[3] a 1 also implicitly assumes that A has flow cookies deployed

If a packet from S to R arrives at A, A checks notices that the sequence of trust relationships along the forward path to R contains at least one "0". A simply forwards the packet along. Similarly, B and C forward the packet along.

When D receives the packet, it notices that there is a continuous trust sequence of 1's, starting from D, all the way to the receiver R; and that D did not trust its upstream neighbor, C. D would also check that the destination prefix, R, was advertised to be protected.

For ISPs such as D to make this inference, we must make minor modifications to the routing tables at their cookie boxes. Assume that the ISP D deploys a distinct cookie box on each of its peering links. The cookie box is similar to a router, with its own routing table etc. The only difference is that each route in the cookie box's table is annotated with the "AND" of the downstream trust sequence, and the inverse of the trust relationship with the upstream. For example, annotation at D for packets from received from ISP C destined for R is:

$(T_{RE}$ AND $T_{ED})$ AND (NOT(T_{CD}))

Where T_{XY} denotes the trust relationship between X and Y. If the annotation is 1, then D does the flow cookie check, insertion or filtering (see below). Otherwise, D lets the packet through. Based on this inference, in the above example, ISP D decides that it must do the flow cookie insertion/check for packets from S to R.

As packets flow past the trust frontier, they are forwarded to the destination. For example, E knows that D can be trusted to vet packets. E simply forwards the packet along. Similarly, R trusts E and accepts the packet as legitimate.

Trust relationships similarly come into play when deciding to filter packets. When R wants to filter packets, it pushes filters upstream to E. E in turn pushes to D, and this happens until the filters hit a "trust boundary" (e.g. the edge between D and C), when the filters are finally installed.

As filters are pushed up the trust chain, tier-1 ISPs may have to install a large number of filters. However, as we described in Section 3.1, filters are mainly needed for connection attempts; attack flows are automatically filtered when the timer for their cookie expires. Therefore, filters can be stored in larger, slower memories since they are only consulted on SYN packets. Further, the IPs can be aggregated in the filters under memory pressure.

4.3 Exploring the Trust Boundary

As described previously, the trust relationships between ISPs implicitly define a region of trust within which a protected server could deploy multiple cookie boxes. We refer to the boundary of this region as the "trust boundary". For example, ISP D above is on the trust boundary

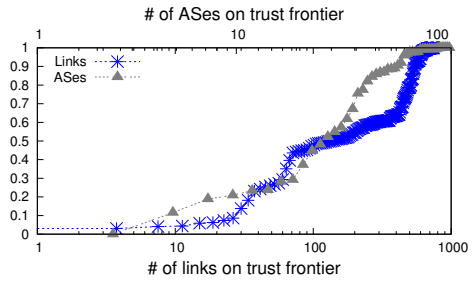

Figure 4: CDF of the number of ASes (top x-axis) and links (bottom x-axis) on the trust boundaries of 13,031 stub networks.

of stub network R for all packets arriving from S. A different ISP may be on the trust boundary for packets from other sources.

As past studies have shown, ISPs higher up in the Internet ISP-hierarchy have higher bandwidth links [5]. Hence, for example, links in ISP D above have higher capacity than those in E. This implies that, independent of the mechanism used, the trust boundary dictates the *maximum* amount of aggregate filtering bandwidth available to a server under existing trust relationships. In addition, ISPs higher up in the hierarchy are known to have a much larger number of interconnections with other ISPs [12]. Therefore, the advantage of using ISPs in the trust boundary over a point deployment for filtering is not only the access to greater bandwidth but isolation between multiple links.

Put another way, the ideal trust boundary should be composed of multiple, large ISPs (tier-1 and tier-2) with routes distributed across boundary links in a manner proportional to the link bandwidth.

In what follows, we explore the properties of trust boundaries on the Internet today using publicly available AS provider-subscriber and peering relationship information inferred from BGP data [19]. Our analysis assumes that trust only transfers across client/provider relationships and ends at peering links. In practice, peering agreements may contain provisions for filtering as well, therefore we consider our analysis pessimistic in the amount of filtering bandwidth available on the trust boundary.

The analysis covers the trust boundaries for 13,031 stub networks. We first look at the total number of distinct ASes and links on the trust boundaries for each of the stub networks (Figure 4). Here, and in the subsequent analysis, we use the term "link" to refer to the collection of several physical interdomain links between neighboring ISPs.

Over 70% of stub networks have 10 or more different ASes on the their boundaries, with over 60 total links. 90% of stubs have 20 or more links on their boundary.

We also look at the distribution of tiers on the trust

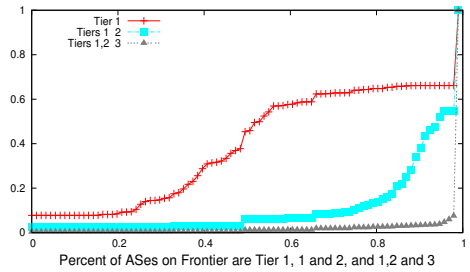

Figure 5: CDF of the percent of ASes on the boundary that are tier 1, tiers 1 and 2 and tiers 1,2, and 3

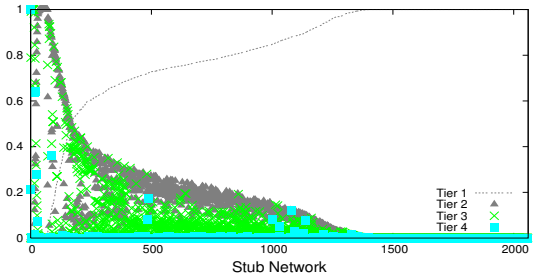

Figure 6: Histogram of the fraction of routes passing through boundary ISPs belonging to tiers 1–4.

boundary. As shown by past studies [5], we assume that links from smaller, or lower-tier ISPs (e.g. tier 3 and 4) are lower bandwidth than those from tier 1 and 2. Figure 5 is a CDF plot of the percentage of autonomous systems that are either tier 1, tiers 1 and 2, and tiers 1,2 and 3. It is clear that the boundary is largely made up of tier 1 and 2 ISPs. The trust boundaries of over 80% of stub networks consist of 85% or more tier 1 or 2 ISPs. This is unsurprising given that larger tier ISPs peer heavily thus ending the recursive passing of trust.

Regardless of the distribution of tiers on the boundary, if many routes to a web servers traverse small ISPs on the trust boundary, then the boundary may still only provide nominal protection. To demonstrate that this is not the case, we analyze routes from all destinations in our data set to each of several stub networks. Figure 6 is a histogram of the percent of routes which traverse each tier (1-4) on the trust boundary for over 8,000 stub networks. As expected, for most trust boundaries, the vast majority of routes pass through tier 1 links. Very few pass through tier 4.

These results suggest that, in most cases, the trust boundary can be used to provide effective, high-speed filtering bandwidth in the network. However, given the nature of peering relationships it is difficult to do a complete study of the trust boundary properties. For example, smaller tier-3 or tier-4 ISPs often engage in private peering relationships. Such relationships cannot be inferred from public BGP data. Another issue that we neglect in our analysis is that the AS level path vector may not necessarily reflect the true forwarding path of a packet (due to internal ISP policies). However, we do believe that our BGP-based scheme and analysis of the trust boundary hold for such situations as well.

5 Related Work

We discussed several related approaches in Section 1. In this section, we elaborate on some of them.

Filter-based approaches, such as Pushback [16] and AITF [7] require the identification and blocking of illegitimate traffic from within the network. In Pushback, a router attempts to identify attack traffic by determining that it is in a congested state, finding an "aggregate" that describes the attack traffic, and pushing the "aggregate" upstream to be blocked close to the source. In contrast, our solution lets the web server discern unwanted traffic.

AITF is based on the observation that a pure filter-based approach such as Pushback requires too much state in core routers. AITF decentralizes the state by pushing filtering rules as close to the source as possible. However, it does not consider how the filtering rules are determined and whether the attack recognition mechanism is resistant against spoofing.

Flow cookies can be viewed as a simplified variant of network **capabilities** [22, 23]. In contrast with capabilities, our scheme offloads connection negotiation to the network and requires no modification to routers, nor to IP or TCP. Also, flow cookies aims for "partial path protection", and trusts one end of the communication: the public web sites; Capability schemes strive to offer "complete path protection".

Firebreak [11] proposes to use IP anycast to redirect traffic to filtering portals close to the source. Flow cookies could be used in conjunction with firebreak to enforce precise, per-flow filtering in a stateless and backwards compatible manner. However, we feel that a more viable deployment strategy is to "push" filtering up from the target rather than expect deployment at the edge, close to the source.

In [18], the authors suggest using virtual nets within the network to filter DDoS. Like our proposal, they suggest deployment close to the victim. Authors in [13] extend this to work over multiple ISPs. However, because the filtering points do not participate in flow negotiation, these schemes are vulnerable to source spoofing and first-packet flooding. Also, we offer a method for dynamically determining where to enforce filtering given a placement of the filtering hardware.

DDoS solutions relying on the deployment of security appliances, such as [2, 3], perform a number of DDoS prevention functions such as offloading the three-way handshake with SYN cookies, enforcing per-flow rate limiting of millions of flows etc. We argue that the end-server must be responsible for distinguishing good vs bad

traffic and pushing the filtering decisions to the network. In addition, we do not require per-flow state lowering the complexity and cost of implementation.

6 Summary

DDoS flooding attacks that target the victim's incoming bandwidth are particularly difficult to contain since the victim cannot directly control the utilization of its incoming link. Several solutions have been proposed both in the industry and research community to tackle this problem. However, these solutions are either inaccurate, ineffective against low-bandwidth floods, impossible to incrementally deploy or require expensive state maintenance.

In this paper, we proposed a simple approach for flood protection that directly addresses the drawbacks of existing mechanism. Our approach builds on current signaling mechanisms (TCP's three-way handshake), past work on network capabilities, and Internet trust relationships (ISP customer-provider relationships).

We outlined flow cookies, a simple, backwards-compatible, point-deployable, network-level protection mechanism that can offer high protection bandwidth to public web servers. Flow cookies helps public web servers implement flexible traffic policies at high bandwidth network locations with minimal additional support from the network.

We argued that the protection offered by flow cookies can be enhanced by deploying flow cookie boxes among ISPs that have a direct or indirect economic relationship with the web server. Such ISPs are said to lie in the web-server's region of trust. We presented a simple BGP-based mechanism to coordinate handshake and filtering activities among multiple such boxes. We study several interesting properties of trust regions in the Internet today.

References

[1] Arbor home page. http://www.arbornetworks.com/.

[2] Netscaler syn flood protection. http://www.netscaler.com/docs/library/ NetScaler-SYNDefense.pdf.

[3] Prolexic home page. http://prolexic,com.

[4] Tcpha home page. http://dragon.linux-vs.org/ dragonfly/.

[5] A. Akella, S. Seshan, and A. Shaikh. An Empirical Evaluation of Wide-Area Internet Bottlenecks. In *Internet Measurement Conference*, Miami, FL, Nov. 2003.

[6] D. Andersen. Mayday: Distributed filtering for internet services. In *USITS, Seattle, WA, 2003.*, 2003.

[7] K. Argyraki and D. R. Cheriton. Active internet traffic filtering: Real-time response to denial-of-service attacks. In *USENIX Annual Technical Conference*, 2005.

[8] D. Bernstein. Syn cookies. http://cr.yp.to/syncookies.html, 1996.

[9] M. Casado, P. Cao, A. Akella, and N. Provos. Flow-Cookies: Using Bandwidth Amplification to Defend Against DDoS Flooding Attacks. Stanford HPNG Technical Report, 2006.

[10] M. Casado and N. McKeown. The Virtual Network System. In *Proceedings of the 36th SIGCSE technical symposium on Computer science education*, 2005.

[11] P. Franics. Firebreak: An ip perimeter defense architecture. http://www.cs.cornell.edu/People/francis/firebreak/hotnets-firebreak-v7.pdf.

[12] L. Gao. On inferring autonomous system relationships in the Internet. 9(6), Dec. 2001.

[13] A. Greenhalgh, M. Handley, and F. Huici. Using routing and tunneling to combat dos attacks. In *Proc. Usenix workshop on Steps to Reducing Unwanted Traffic on the Internet*, July 2005.

[14] A. Keromytis, V. Misra, and D. Rubenstein. Sos: Secure overlay services. In *Proceedings ofACM SIGCOMM'02*, 2002.

[15] S. Lee, J. Lui, and D. Yau. Admission control and dynamic adaptation for a proportional-delay diffserv-enabled web server. In *SIGMETRICS '02*, pages 172–182, New York, NY, USA, 2002. ACM Press.

[16] R. Mahajan, S. M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker. Controlling high bandwidth aggregates in the network. *SIGCOMM Comput. Commun. Rev.*, 32(3):62–73, 2002.

[17] W. B. Norton. Internet service providers and peering. In *Proceedings of NANOG 19*, Albuquerque, New Mexico, June 2000.

[18] R. Stone. Centertrack: An ip overlay network for tracking dos floods. In *In Proceedings of the Ninth USENIX Security Symposium*, August 2000.

[19] L. Subramanian, S. Agarwal, J. Rexford, and R. H. Katz. Characterizing the Internet Hierarchy from Multiple Vantage Points. June 2002.

[20] A. B. Tadayoshi Kohno and K. Claffy. Remote physical device fingerprinting. In *IEEE Symposium on Security and Privacy*, 2005.

[21] M. Welsh and D. Culler. Adaptive overload control for busy internet servers. In *USITS*, 2003.

[22] A. Yaar, A. Perrig, and D. Song. Siff: A stateless internet flow filter to mitigate ddos flooding attacks. In *In Proceedings of the IEEE Security and Privacy Symposium*, 2004.

[23] X. Yang, D. Wetherall, and T. Anderson. A dos-limiting network architecture. In *Proc. ACM SIGCOMM*, 2005.