Social Security for the Internet

Thomas E. Anderson and Arun Venkataramani
DRAFT: DO NOT REDISTRIBUTE

Abstract

Recent times have seen several intriguing proposals to combat denial-of-service (DoS) attacks, where malicious users flood a limited resource preventing legitimate users from accessing that resource. All of these proposals reconcile themselves to the following common fate: if B bad users attack a limited resource being accessed by G good users, then good users suffer an $O(\frac{B}{B+G})$ degradation in performance. Unfortunately, a common case of $B \gg G$ makes this an unacceptable state of affairs. In this paper, we present SSN, a "social security network", as a building block for a secure Internet architecture with the following properties. First, B bad users can degrade the performance of good users by at most $O(\frac{B}{N})$ where N is the total number of users in the network, and by $O(1)$ in the common case. Second, our proposal is readily implementable with today's technology and expects only minimal router support in the form of *capabilities*, a mechanism widely believed to be necessary to address DoS.

1 Introduction

Securing network resource allocation is one of the most important architectural issues that the Internet faces today. The Internet architecture makes it easy to launch denial-of-service (DoS) attacks, where a collection of hosts with sufficient bandwidth can disrupt communication between arbitrary hosts simply by flooding the network with unwanted packets. The inevitability of operating system vulnerabilities gives DoS a a high damage-to-effort ratio compared to most traditional infrastructural services, e.g., a few lines of code [13, 5] can cause millions of dollars of damage while allowing the attacker to escape without a trace.

Recent times have seen a plethora of proposals for addressing DoS based on whitelists [10, 22], blacklists [6, 8], currency [12], authentication and admission control [9], defense by offense [20] and so on.

We take a fresh look at securing network resource allocation in the light of the following observation. All existing proposals reconcile themselves to a common fate, namely, if B compromised (bad) hosts attack a limited resource being concurrently accessed by G legitimate (good) hosts, then good hosts suffer an $O(\frac{B}{B+G})$ degradation in performance. Unfortunately, OS vulnerabilities render $B \gg G$ a common case making this degradation an unacceptable state of affairs.

Our position is that a carefully designed network architecture can and should limit the impact of bad nodes on *any* good node commensurate to the fraction of bad nodes relative to *all* nodes in the network. This position is similar in spirit to the design of byzantine fault-tolerant (BFT) distributed systems. For a further analogy, even in the real-world, security depends as much on the everyday vigil maintained by all citizens as well as preventive law enforcement mechanisms.

To this end, we propose SSN, a social security network, that limits the impact of bad nodes proportional to the fraction of bad nodes in the entire Internet when there exists means to distinguish between bad and good nodes. To appreciate this claim, observe that it is easy to create a botnet of a 100K hosts that targets a Web server being concurrently accessed by 10 good users, which can cause considerable damage to good users even though 100K is a small fraction of hosts in the Internet and reverse-Turing tests can be used to distinguish bots from good users (see Section 2 for details).

SSN accomplishes this objective using the following key ideas. First, it assumes and builds upon network capabilities [22] that require a host to explicitly obtain permission to send packets along a network path. Second, it uses a novel social overlay network that rapidly creates additional channels of communication available only to good hosts. Adjacent hosts or "friends" in this social network acquire capabilities a priori to sustain communication under attack. Friends help each other through two novel mechanisms: (a) delegable authentication, and (b) control channel offense. The former allows authentication or acquisition of capabilities to be delegated to other nodes enabling defense-in-depth, while the latter is similar in spirit to a recent proposal advocating "offense" for DDoS defense, but applies it in a careful and controlled manner to the capability acquisition channel. Third, it uses a careful division of labor between routers and end-hosts to allow for a minimal and evolvable security architecture. Capabilities are largely believed to be necessary to address the DoS problem. SSN delegates all other responsibility to the overlay network of end-hosts.

The rest of this paper is organized as follows. Section 2 describes in detail relevant related work. Section 3 presents assumptions, design principles, and key components of SSN's architecture, and analysis of its security. Section 4 presents a broader discussion and open questions that must be addressed to make deployment of SSN practical.

2 Background and Related Work

SSN heavily leverages ideas in previous proposals to address DoS. We first discuss these proposals to illustrate both their strengths and weaknesses, and how SSN builds upon them. As discussed in [4], we categorize proposals as prevention detection and recovery, resilience, and deterrence mechanisms.

Prevention Prevention is the baseline defense mechanism that is most critical to determine an architecture's effectiveness to DoS attacks. Several recent *whitelist* approaches use access control to limit resource access only to legitimate users that can prove their legitimacy to the provider of the resource. These can be classified as overlay-based filtering or network capabilities supported in routers.

Overlay-based approaches like SOS [10] and Mayday [1] allow a destination to limit communication based on previously established communication. In both approaches, an offline authenticator such as a large overlay verifies the legitimacy of an incoming packet and adds a secret to it before forwarding it to the destination. Downstream routers discard all packets that do not contain the secret. In contrast, SSN's goal is to allow DoS-tolerant communication between arbitrary hosts without prior arrangement as well as exploit such arrangement when feasible. For example, it is reasonable to limit access to human network administrators when a large-scale disaster happens, but arbitrary hosts should be able to communicate with each other under normal circumstances.

TVA [22, 2], or Traffic Validation Architecture, proposes *network capabilities* as a necessary mechanism to address DoS. Their disarmingly simple insight is that the Internet fundamentally lacks a mechanism that allows a destination or a router to refuse packets from a host. Their proposal addresses this deficiency by empowering routers and destinations to grant fine-grained capabilities (e.g., permission to send 10 packets in the next 5 seconds) to hosts they think are legitimate. Each router and the destination independently verifies the capability carried inside a packet and discards the packet if the verification fails. To improve deployability, only routers at administrative or "trust" boundaries are required to support capabilities.

The acquisition of capabilities itself is a delicate task that TVA accomplishes using an open *control channel* that hosts use to request capabilities, and a *data channel* that allows only packets with verifiable capabiltities. A host sends a capability request via the control channels of the capability routers along the path, and each of these routers and the destination inserts a verifiable secret into the request packet that is returned to the host. The host can subsequently send data packets containing these secrets or capabilities through a separate *data channel* (pinned to the same sequence of capability routers) till the capabilities expire. By limiting the bandwidth of the control channel to a small fraction of the total bandwidth (their suggested value is about 5%), the bulk of the bandwidth can be made accessible only to legitimate users. Furthermore, once a capability is granted, legitimate users can renew capabilities from inside the data channel without having to compete again for access to the control channel.

Although an authentication mechanism is fundamental to limit resource accesss to legitimate hosts, both overlay-based approaches as well as network capabilities described above shift the DoS problem from the resource being protected to the authentication channel that by design is open to arbitrary hosts. Thus, DoS manifests itself as a *resource acquisition latency* (RAL) as opposed to degraded allocation of resources. For example, if 100K bad hosts compete for a limited resource, say a Web server, with 100 good hosts, a good host on average will take 10^3 times as long to acquire a capability, simply because the control channel is open to all hosts. The same is true at a bottlenecked capability router in TVA.

Detection and Recovery Detection and recovery mechanisms attempt to detect aberrant hosts and punish them. These include ingress-filtering [6, 14] to discard spoofed packets at the edge of the network, or traceback [16, 17] mechanisms to identify the origin of attack packets, and pushback [8, 3] mechanisms to reactively filter attack packets in-network before they reach the protected resource. With massive-scale DDoS attacks using botnets [9] and viruses becoming commonplace, ingress-filtering is insufficient to prevent denial-of-service attacks as the attacking hosts use their real IP addresses. Traceback and pushback are insufficient for two reasons. First, they are too late to limit RAL, and second, they are often error-prone and must trade-off precision of detection with scalability and the risk of stifling legitimate traffic. Statistical techniques for anomaly detection also suffer from a similar problem. All of these arguments have also been recognized by others [22, 20].

Botz-4-Sale developed by Kandula *et al.* [9], is a Web server architecture that combines prevention, detection, and recovery. The key idea is similar in spirit to capabilities implemented only at a Web server, but with the following key differences. First, the authentication step is enforced only when a DoS attack is suspected. Second, the authentication mechanism is more sophisticated and may use CAPTCHAs [19] or other forms of puzzles or currency. Third, and most importantly, the fraction of bandwidth allocated to the authentication channel, unlike TVA, is not a fixed fraction of the total bandwidth, but is adjusted dynamically based on a control algorithm that attempts to maximize the goodput of the server. Admission control by randomly dropping authentication requests is used to control the fraction of bandwidth allocated to the authentication channel.

Kandula *et al.* show that the optimal goodput of the server is roughly $O(\frac{\lambda_s}{\lambda_s+\lambda_a})$ where λ_s and λ_a are rates of legitimate and attack requests respectively. So, a targeted DDoS attack will cause a degradation in goodput proportional to the number of attacking hosts. To counter this problem, Botz-4-Sale remembers IP addresses of unresponsive hosts that do not answer CAPTCHAs for a long time and discards packets from them at the MAC layer. However, such a reactive mechanism is ineffective for limiting RAL as the damage is already done by then. Furthermore, dropping packets based on IP address can penalize good hosts unless a safe mechanism to verifying identities is in place. We contend that, in general, a safe mechanism to verify identities will induce non-negligible work at a server, thereby necessitating a network-wide mechanism to prevent attack packets from reaching the server in the first place.

DDoS defense by offense [20] is a refreshingly bold proposal by Walfish *et al.* that does not rely on distinguishing between good and bad users. Instead, it advocates that when a DoS attack is suspected, good users should blast packets into the network as fast as they can to compete with bad users on an equal footing. Although such a response will certainly benefit good users more than no response at all, the proposal itself appears impractical for immediate deployment. First, detecting whether a DoS attack is happening is error-prone. Second, offense can significantly increase the bandwidth bills of good users and increase congestion in the network. Third, resource allocation is unfair even between good users as offense results in hierarchical link-sharing [7]. Therefore, if a lone good user is stuck with several bad users on a path with multiple bottlenecks, it can get an arbitrarily small fraction of the bottleneck bandwidth. Finally, offense as proposed suffers from at least an $O(\frac{B}{B+G})$ degradation in allocated bandwidth even when there is just a single bottleneck.

Resilience and Deterrence Resilience mechanisms include supporting alternate paths [18, 21] to circumvent network bottlenecks created by attacking hosts, massive-scale replication and capacity provisioning for critical services, hiding the server [11] and so on. Deterrence mechanisms can be aided by auditing, forensic analysis, and legally enforced penalties to discourage would-be attackers. Although such mechanisms have been somewhat useful in discouraging extortion attacks [5], in general, resilience and deterrence are believed to be impractical or insufficient to address DoS [4].

3 SSN Architecture

3.1 Assumptions and Design Principles

Our attack model is powerful, but the security guarantees we claim are based on certain assumptions about the severity of the attack.

First, we assume that OS vulnerabilities are inevitable and may let an attacker gain control of a large number of nodes (a *node* refers to a router or host). However, the fraction of compromised nodes is a small fraction of the total number of nodes in the network within any given *window of vulnerability*. For example, a worm can rapidly create a botnet of several hundred thousand machines, but that is still a small fraction of the total number of hosts in the Internet. The window of vulnerability refers to the expected time to patch an operating system vulnerability. This assumption is fairly common in byzantine fault-tolerant distributed systems and appears reasonable for the Internet. If a significant fraction of the entire Internet is compromised, then any architecture is unlikely to be effective. Nevertheless, we allow the ratio of compromised nodes to the number of good nodes attempting to access a specific resource — a Web server; or computation, memory, or link bandwidth at a router — to be high.

We assume that each node has a test to determine if a host is good or bad. (Note that if there is no way to distinguish between good and bad users, fair-queueing of resources is the best possible defense and incurs a $O(\frac{B}{B+G})$ degradation.) In practice, hosts have some means to distinguish between good and bad hosts such as a login and password, or credit card information, or CAPTCHAs [19], or high priority accounts and so on. We assume that the amount of resource required to administer this test is non-negligible. However, we do assume that it is a small fraction of the resource that will be granted upon success of the test. Finally, bad nodes may collude amongst themselves and send as much data to each other as the network allows them. We will refer to these attacks as *indirect attacks*.

We assume that there exists an underlying routing protocol that computes point-to-point (unipath) routes much like the Internet today. Furthermore, we assume that the routing protocol is byzantine fault-tolerant, i.e., the number of routers compromised in any window of vulnerability is sufficiently small that the routing layer can offer an abstraction of point-to-point connectivity between any two end-hosts along a path consisting only of good routers. Nevertheless, we do not assume any additional trust between routers or between routers and end-hosts. The essential assumption here is the separability of the routing protocol (that provides reachability) from the bandwidth allocation mechanism.

Finally, we make standard cryptographic assumptions, in particular, the existence of irreversible one-way hash functions. However, we do not assume the existence of a public-key infrastructure in the design of SSN.

The following principles guide the design and implementation of SSN.

1. *Minimal network layer*: An architecture should impose minimal new functionaility into routers to (i) keep

the network layer simple, and (ii) avoid introducing new security bugs. SSN assumes that routers at trust boundaries support capabilities, a mechanism deemed minimally necessary to address DoS attacks.

2. *Only trust thyself*: Routers and end-hosts should minimize trust in other nodes. A router in SSN only verifies secrets generated by itself as in TVA. In particular, other routers or end-hosts are not permitted to install filters at an upstream router to block traffic from a host as such a mechanism can be used to launch attacks based on exhausting router state or maligning innocent senders.

3. *Nip the evil in the bud*: An architecture must restrict the resources consumed by attack traffic by squelching it as close to its source as possible. We show how SSN reconciles this principle with the previous one.

4. *Flexible control*: No end-host is guaranteed good or bad. A destination should be able to dynamically define what goodness of a node sending packets to it means. For example, under critical situations, only privileged human users may be allowed access, while at other times, any valid login and password may suffice.

5. *Penny wise and pound wise*: A complete architecture must carefully account for bandwidth, computation, and memory overhead it induces for both control and data traffic. For example, a single control packet from each of 100K bad users can render a control channel inacceessible to good users.

3.2 SSN Architecture

At the core of SSN, is a social overlay network on top of an underlying capability-based network layer. The social network helps distribute the impact of targeted attacks using three key ideas: (i) limited pre-acquired capabilities, (ii) delegable authentication, and (iii) creating channels of communication available only to good nodes. We elaborate on each of these next.

Each host picks a small number of other hosts at bootstrap time as its "friends". Friends acquire the capability to send a small number of packets to each other and keep refreshing this capability at a low rate from inside the corresponding data channel. The control channel is open to all hosts as in TVA. Routers perform fair-queuing of capability requests based on coarse-grained path identifiers, while legacy packets are processed using a low priority queue.

Path diversity When a host encounters an in-network bottleneck at a capability router, it uses any overlay path in SSN leading to the destination to contact the destination. The host sends its capability request to the destination tunneled through the data channel in the overlay path. Since the overlay path has acquired capabilities a priori, the tunneled packets do not encounter additional latency in getting the first packet through to the destination. Thus, good hosts can circumvent in-network bottlenecks that bad hosts create by flooding the control channel of a capability router.

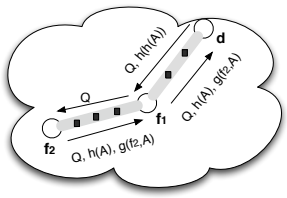

Figure 1: Example of delegable authentication. Black squares represent capability routers and white circles represent SSN hosts.

Delegable authentication Paths in the social network are available to bad hosts as well, so enabling path diversity by itself is not sufficient to counter flooding of the control channel. SSN uses the notion of delegable authentication that lets a destination delegate the task of verifying the legitimacy of a host wishing to send packets to it to another host. Without loss of generality, we assume that this test is in the form of a question and an answer such that only good nodes can answer the question correctly.

Let Q and A denote the question and answer respectively. As shown in Figure 1, the destination d sends its immediate friend f_1 the question Q and $h(h(A))$ where h is a one-way hash function also sent to f_1. Friend f_1 can now administer this test to its friend f_2 which is the host that initiated communication to d. If f_2 is good, it and returns $h(A)$ and $g([A, f_2])$ to f_1 where g is another one-way hash function and $g([A, f_2])$ is the hash of the answer concatenated with f_2's network identifier such as its IP address. Now, f_1 can verify that f_2 is good using $h(A)$ and send $g([A, f_2])$ to d that can also verify that f_2 is good. If the verification at f_1 fails, then it does not report anything to d. Thus, d only gets capability requests from hosts that have been verified to be good by its friends. This example involved only a path of length two, but is easily generalized to longer paths, where f_1 can recursively delegate the test to f_2 and so on.

The key to limiting bad hosts from using paths in SSN is that a host reports only successful administrations of the test to its parent along SSN closer to the destination. Furthermore, immediate friends of d such as f_1 retrieve at most one test from d based on requests initiated by bad upstream hosts. f_1 reuses this test for subsequent requests to contact d until some host correctly answers the test. If we assume the number of friends of any host to be at most a small constant c, bad hosts can at most force d to send a total of c useless tests along SSN.

The destination d processes capability requests tunneled via its friends on a priority basis as it knows they will be successful. However, a friend f_1 could get compromised and start forwarding spurious requests to d. In this case, d can simply stop accepting packets from f_1's queue and cease to renew their relationship.

Control channel offense The delegable authentication mechanism described above assumes that d's friends (i) have adequate bandwidth to tunnel capability requests and subsequent data packets to d via SSN, and (ii) have not been themselves compromised. While destinations can increase the likelihood of satisfying the two requirements by carefully choosing friends, e.g., popular reputed servers can choose similar servers as friends, in general, one or both of the conditions may not hold.

In this case, SSN hosts use a more powerful *offense* mechanism by requesting requesting their friends to use the underlying network path to send capability requests to d at the maximum rate allowed by the control channel. The consequence is that the control channel at the destination is flooded by authentication requests from B bad users and a large, comparable number of good users G' in SSN. Furthermore, d uses admission control by randomly dropping authentication requests to limit the capacity of the control channel similar to [9]. If $G' > B$, the consequence is that bad users on average double the latency to get the first packet through to the server. The basic idea of offense to increase the number of good users is inspired by [20], with the following key differences that make this form of offense practical: first, offense is invoked only on a small fraction of total bandwidth that can be independently controlled by each capability router as well as the destination; second, offense is invoked only when path diversity and pre-acquired capabilities in SSN are unavailable. Note that each host and destination can track the goodput of its control channel, i.e, how many capability/authentication requests are successful to track the level of bad users in the network. Thus, unlike [20], where a false suspicion of DoS can cause considerable harm, offense in SSN is safe.

Data channel reuse If a host a encounters a friend b enroute to the destination d along SSN such that b already has an open data channel to d, then a's authentication request can be tunneled via that data channel exiting SSN early. As before, subsequent capability to send data is granted to a only upon successful authentication of a.

Tracking failures Friends track failed authentication attempts by upstream friends and cease to accept more tunneling requests from them to that destination. Furthermore, if a friend makes several failed authentication requests to different destinations, then the existing relationship is not renewed when the current low-rate capability expires. This aspect is similar in spirit to pushback, with the following key differences: first, per-flow state is stored only at hosts and not at routers; second, a host stores state corresponding to a friend only when it suspects the friend to be bad, not at the bequest of some othe node. Similar to [9], the per-flow state can be approximately tracked using a bloom filter that can be sized in accordance with available storage at a host. Note that tracking per-flow state at capability routers can not exploit optimizations used by TVA to track only heavy senders as all senders in the control channel are subject to the same fair-queued rate. Finally, per-flow state must be stored as opposed to per-sender state as, otherwise, a compromised destination can implicate a good user by rejecting its authentication attempts.

3.3 Security and Overhead Analysis

In Section 2, we showed how existing proposals are limited by an $O(\frac{B}{B+G})$ degradation factor. In contrast, SSN spreads the impact of targeted DoS attacks to different regions of the network. To quantify the storage overhead induced by SSN setup, consider that each host sets up a small constant number of friend capabilities. Routers store state corresponding to these capabilities only when the connections are active. Even then, the avergae state stored by a router is $O(k \cdot H/R)$ where H and R are the total number of hosts and routers respectively and k is the maximum number of friends allowed. In the typical case, when path diversity suffices to circumvent in-network bottlenecks, the degradation in RAL for any host is at most $O(1)$ assuming a random distribution of attacked hosts. When control channel offense is invoked, the degradation in RAL is again $O(1)$ as the number of good users and therefore the goodput matches the wasted work in trying to authenticate bad hosts. In both cases, we assume that B is a vanishingly small fraction of the total number of nodes in the network.

4 Discussion and Open Questions

SSN addresses a fundamental limitation of network capabilities, namely, that by exposing an open control channel, the DoS problem shifts from resource allocation to resource acquisition latency. Furthermore, SSN is an integrated proposal that lets each node — a capability router or destination — dynamically apportion bandwidth between its control and data channels to optimize the goodput of the data channel. Finally, when all else fails, SSN falls back on a controlled form of offense on the control channel. While we believe that all of these components are essential to designing a secure Internet architecture, our preliminary proposal has left several questions unanswered.

First, we have only outlined mechanisms, but left the exact control algorithms and their parameters underspecified. For example, unlike the analysis in Botz-4-Sale that assumes a fixed attack rate and legitimate request rate, these rates are changing when control channel offense is invoked making the bandwidth partitioning much more challenging. Second, the fan-out used in offense to increase the strength of good nodes to match that of bad nodes is also a control parameter that must be carefully estimated. Third, we need to quantify the effect of increased resource consumption due to inflated

tunneled paths via SSN till the underlay path manages to acquire an end-to-end capability. Theoretical and experimental analyses of these questions is ongoing work.

SSN fundamentally assumes the presence of a test that nodes can use to protect themselves from bad hosts. While this is often the case, especially, for critical resources, in general, such a test may not exist. In this case, SSN allows B bad hosts to inflict a performance degradation of $O(\frac{B}{B+G})$ upon G concurrent good hosts. Without distinguishability, the best a network can do is to allocate resources fairly as bad hosts can acquire a fair share of network resources simply by behaving like good hosts.

Although path diversity and offense can help alleviate indirect attacks, SSN does not offer a complete solution to restrict indirect attacks. Note that indirect attacks are equivalent to indistinguishability as there is no way to determine that two hosts sending data to each other in collusion are bad. Thus, as above, the best a network can do is to allocate a fair share of control or data channel bandwidth along the corresponding paths. On the positive side, it is difficult to compromise a large number of strategically placed nodes.

SSN assumes that authentication tests are delegable. We described how to delegate tests that can be represented as a question and answer. We believe that several commonly used tests such as a login and password, CAPTCHAs, shared secrets known only privileged users or conveyed out-of-band can be cast as a question and answer. However, other forms of authentication such as allowing only hosts from a specific geographic region to access a resource may require more network support.

SSN assumes the presence of an underlying routing layer that provides reachability even in the presence of a small fraction of compromised routers. We note that Perlman's dissertation [15] from almost two decades back continues to be the most comprehensive design of a byzantine fault-tolerant (BFT) routing layer. However, a practical implementation of a BFT routing layer is still an open question as routers must fundamentally balance the increased security of flooding with the increased overhead. Furthermore, asynchrony makes it difficult to distinguish between router failures and legitimate delays. SSN assumes that it is feasible to design a low-overhead routing layer that provides reachability on top of which it provides a practical resource allocation architecture.

There are several auxiliary benefits to using social networks for security. First, a key trait of social networks is that it clumps together hosts that trust each other and also have potentially overlapping interest in content. Thus, hosts in SSN can also obtain cached static content enroute to the destination from friends. Second, "friends won't let friends launch DoS attacks" if they can determine that the friend is infected based on a virus signature. Furthermore, they can supply a patch to an OS vulnerability of they have procured one, or can inform them offline of the vulnerability. Third, hosts can intelligently choose friends to ensure OS diversity among friends similar in spirit to N-version programming in byzantine fault-tolerant distributed systems. Although SSN does not rely on any such trust assumptions, the "falling in place" of these benefits is encouraging.

References

[1] D. Andersen. Mayday: Distributed filtering for Internet services. In *USITS*, 2003.

[2] T. Anderson, T. Roscoe, and D. Wetherall. Preventing internet denial-of-service with capabilities. *SIGCOMM Comput. Commun. Rev.*, 34(1):39–44, 2004.

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

[4] S. Bellovin, D. Clark, A. Perrig, and D. Song. A Clean-Slate Design for the Next-Generation Secure Internet, July 2005.

[5] FNI busts alleged DDoS mafia. http://www.securityfocus.com/news/9411.

[6] P. Ferguson and D. Senie. Network Ingress Filtering: Defeating Denial of Service Attacks that Employ IP Source Address Spoofing. Technical Report RFC-2475, IETF, June 1999.

[7] S. Floyd and V. Jacobson. Link-sharing and resource management models for packet networks. *IEEE/ACM Trans. Netw.*, 3(4):365–386, 1995.

[8] J. Ioannidis and S. M. Bellovin. Implementing pushback: Router-based defense against DDoS attacks. In *NDSS*. The Internet Society, February 2002.

[9] S. Kandula, D. Katabi, M. Jacob, and A. W. Berger. Botz-4-Sale: Surviving Organized DDoS Attacks That Mimic Flash Crowds. In *USENIX NSDI*, May 2005.

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

[11] K. Lakshminarayanan, D. Adkins, A. Perrig, and I. Stoica. Taming ip packet flooding attacks. *SIGCOMM Comput. Commun. Rev.*, 34(1):45–50, 2004.

[12] D. Mankins, R. Krishnan, C. Boyd, J. Zaho, and M. Frentz. Mitigating distributed denial of service attacks with dynamic resource pricing. In *Annual Computer Security Applications Conference (ACSAC 2001)*, 2001.

[13] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. The Spread of the Sapphire/Slammer Worm. http://www.cs.berkeley.edu/ nweaver/sapphire/, January 2003.

[14] K. Park and H. Lee. On the effectiveness of route-based packet filtering for DDoS attack prevention in power-law internets. In *ACM SIGCOMM*, pages 15–26, 2001.

[15] R. Perlman. *Network layer protocols with byzantine robustness*. PhD thesis, MIT LCS, 1988.

[16] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Practical network support for ip traceback. *SIGCOMM Comput. Commun. Rev.*, 30(4):295–306, 2000.

[17] A. C. Snoeren, C. Partridge, L. Sanchez, and C. Jones. Hash-based ip traceback. In *ACM SIGCOMM*, pages 3–14, 2001.

[18] Stefan Savage et al. Detour: A Case for Informed Internet Routing and Transport. *IEEE Micro*, 19, No. 1:50–59, January 1999.

[19] L. von Ahn, M. Blum, N. Hopper, and J. Langford. Captcha: Using hard ai problems for security. In *EUROCRYPT*, pages 294–311, 2003.

[20] M. Walfish, M. Vutukuru, H. Balakrishnan, D. Karger, and S. Shenker. DDoS Defense by Offense. In *ACM SIGCOMM 2006*, Pisa, Italy, September 2006.

[21] X. Yang. Nira: a new internet routing architecture. In *ACM SIGCOMM FDNA*, pages 301–312, 2003.

[22] X. Yang, D. Wetherall, and T. Anderson. A DoS-limiting network architecture. In *ACM SIGCOMM*, pages 241–252, 2005.