The Case for TCP/IP Puzzles

Wu-chang Feng
OGI@OHSU
wuchang@cse.ogi.edu *

ABSTRACT

Since the Morris worm was unleashed in 1988, distributed denial-of-service (DDoS) attacks via worms and viruses have continued to periodically disrupt the Internet. Client puzzles have been proposed as one mechanism for protecting protocols against denial of service attacks. In this paper, we argue that such puzzles *must* be placed within the slim waistline of the TCP/IP protocol stack in order to truly provide protection. We then describe several scenarios in which TCP/IP puzzles could be used to thwart port scans and coordinated DDoS attacks. Finally, while puzzles hold the promise of being able to change the Internet landscape, we describe a large number of open research issues that must be resolved before such a vision can be achieved.

1. INTRODUCTION

Upon the spread of the Morris worm in 1988, Dave Clark relates a story of a call he received from an angry program manager asking for an explanation for why this was possible and what the program manager should tell his superiors [1]. The response was that the Internet did exactly what it was designed to do, spread the worm as quickly and as efficiently as possible. In fact, the fast spread of the worm demonstrated the strength of the Internet's design. The program manager replied that such an excuse would work this time, but that there had better be a better answer the next time.

Twenty years have now passed and unfortunately, there is no better answer. As evidenced with the Nimda, Code Red, and SQL Slammer [2, 3, 4, 5, 6] worms, the Internet still efficiently spreads worms and viruses. As recently reported [7], the SQL Slammer worm, managed to claim the dubious title of being the first "Warhol" worm, a worm able to lay waste to a significant portion of the Internet in under 15 minutes.

There are many underlying reasons that allow distributed denial of service (DDoS) attacks to occur. From faulty software to a network layer that allows the sender to fill in its own address, a confluence of architectural design features and implementation problems have given hackers the tools necessary to destroy what so many in the networking community have worked so hard to build. In direct contrast to the Internet is the phone network, which does not often suffer from malicious distributed denial of service attacks perpetrated by a handful of rogue hackers: something that is relatively common with the Internet. While there are many reasons for this, two architectural reasons in particular that help include a stronger binding of end-points to their addresses (phone numbers) and the circuit-based service paradigm that requires a large number of connections be made in order to perform a resource-based attack. An effective DDoS attack would require the hacker to physically knock a large number of phones off their hooks.

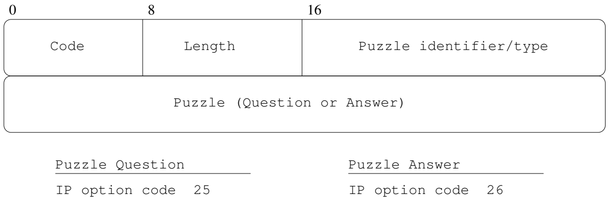
In order to provide Internet services with the same resiliency to resource depletion attacks as the phone network, client puzzles [8, 9, 10, 11, 12, 13, 14] have been proposed as a mechanism for pushing work back onto clients requesting service. With client puzzles, a server or network being protected generates a cryptographic puzzle that a client must answer correctly before it is given service. Such a mechanism gives servers and the network the ability to selectively push back load to the source of an attack when overloaded. When done correctly, this achieves for the Internet, what already exists for the phone network: a requirement that malicious attackers must co-opt an extremely large number of clients in order to perform resource-based attacks.

We argue that the ability to push load selectively back to clients must be implemented and deployed within the slim waistline of the TCP/IP protocol stack in order to effectively deal with DDoS attacks.

In the spirit of efforts to re-visit the Internet's architecture [15], we are exploring a change to the fundamental service paradigm of the Internet in order to allow those providing service to arbitrarily push load back on those requesting service. Although, it is an open research issue, such a change could significantly impact the ability to successfully launch

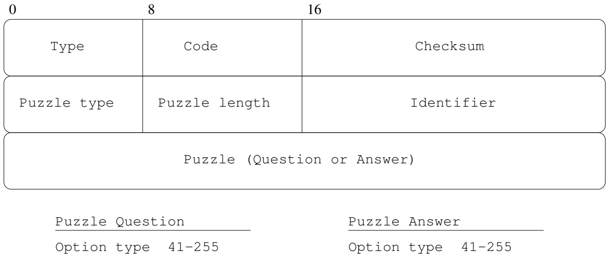
*This material is supported in part by the National Science Foundation under Grant ANI-0230960 and the generous donations of Intel Corporation. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation or Intel.

ACM SIGCOMM 2003 Workshops August 25&27, 2003, Karlsruhe, Germany



Puzzle Question		Puzzle Answer	
IP option code	25	IP option code	26

Figure 1: An IP puzzle option



Puzzle Question		Puzzle Answer	
Option type	41-255	Option type	41-255

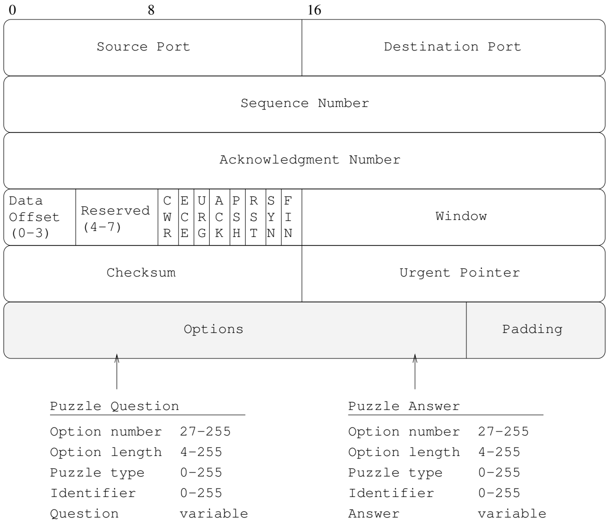
Figure 2: An ICMP puzzle option

denial of service attacks. In particular, by using the willingness and ability for clients to selectively do work on behalf of the server as a crude indication of "intent", such a mechanism can slow down the spread of worms and viruses as well as force hackers to compromise many more machines in order to successfully launch denial-of-service attacks.

2. TCP/IP PUZZLES

Client puzzles provide a capability within the network and end systems to mitigate denial of service attacks. We argue that such functionality *must* be placed in the basic network (IP and ICMP) and transport (TCP, SCTP, UDP) layers. The design of these layers has, in part, been motivated by the end-to-end principle in which underlying protocols do not implement functionality that can and must be implemented by higher-level, end-to-end protocols. This design principle pushes out complexity to the end-hosts while preserving a clean, simple network design.

While the end-to-end argument has motivated the design of a slim network and transport layer, the argument also motivates the need to put puzzles within the waistline. The key is that denial-of-service activity can happen at any layer and only needs to break one link in the end-to-end chain in order to be successful. Because of this, the ability to push back against participants of a DDoS attack must be placed at a layer that is shared across all applications. For example, DoS-resistant authentication protocols [12, 10] are helpless against an IP flooding attack. In general, application-level puzzles only afford protection when the attacker employs only that particular protocol in the attack.



Puzzle Question		Puzzle Answer	
Option number	27-255	Option number	27-255
Option length	4-255	Option length	4-255
Puzzle type	0-255	Puzzle type	0-255
Identifier	0-255	Identifier	0-255
Question	variable	Answer	variable

Figure 3: A TCP puzzle option

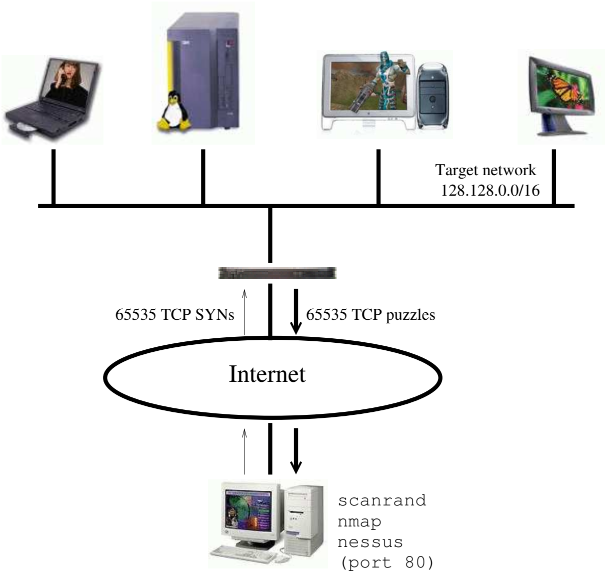


Figure 4: The push-back firewall

Since the only chance there is to push back an arbitrary DoS attack is to put protection at layers that are common to all Internet activity, we believe that the strength of client puzzles will only become significant when placed in common network and transport layers. For example, in the network layer, by adding two new types of messages within ICMP [16] or IP [17], a router or firewall could pass puzzles to its upstream nodes that force the node to compute something in order to maintain its current level of IP forwarding service. In the transport layer, by adding two new TCP options [18], end-hosts could force clients to do the same. Figures 1, 2, and 3 show examples of protocol additions to IP, ICMP, and TCP that could be used to support puzzles. The examples use the currently available IP option, ICMP type, and TCP option numbers. Similar changes would be possible for the other protocols listed above.

3. SCENARIOS

3.1 The push-back firewall

One of the functions enabled by TCP/IP puzzles is the ability to thwart the probing being done by hackers and

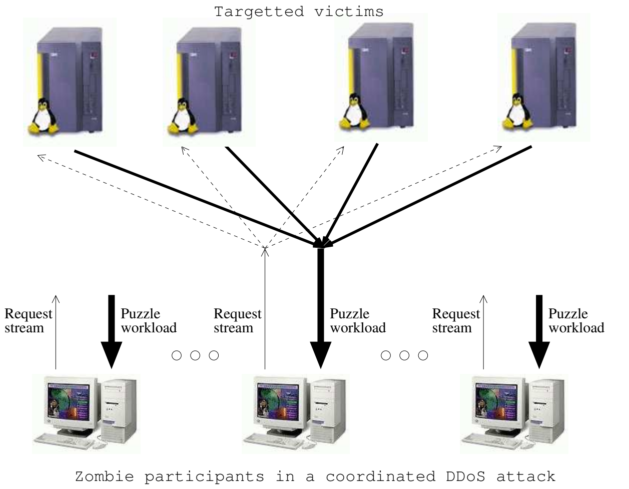


Figure 5: Mitigating coordinated DDoS attacks

worms in order to discover and compromise vulnerable systems. For hackers, slowing down the information gathering significantly increases the time they require to find vulnerable machines using tools such as `nmap` [19], `nessus` [20], and `scanrand` [21]. For worms and viruses, slowing down this process can completely determine their impact. There have been anecdotal cases where simple tweaks in parameterization and simple tweaks in algorithms for finding victims completely altered the behavior of a DDoS attack [2, 22].

One application for TCP/IP puzzles would be to build into firewalls, the ability to respond to all requests with puzzles that the client must solve before being allowed connectivity information. We term such a firewall a "push-back" firewall. Rather than push a filter back into the network such as related approaches [23, 24], the push-back firewall instead pushes CPU load in the form of a puzzle directly back onto clients before they are given any further information. Consider the class B network shown in Figure 4. It has been shown that a single instance of the `scanrand` tool can probe an entire class B network (65536 IP addresses) within seconds [21]. In one test, the tool found 8300 web servers running inside a multinational corporation's class B network in 4 seconds [25]. `scanrand`'s power lies in the fact that it uses a notion of inverse SYN cookies to remain stateless and only requires one socket to do the probing. Consider the same probing done against a push-back firewall and network servers that return puzzles in response to all TCP SYN packets regardless of whether or not a service is available at that location. If each puzzle consumed only one second of CPU time for the attacker's machine to solve, the same probe would take over 18 hours to run. While not a panacea, the ability to selectively slow down hackers and worms via puzzles can add a significant layer of protection to the Internet's infrastructure today.

3.2 Coordinated DDoS

Coordinated distributed denial of service attacks (DDoS) can be used to simultaneously bring down multiple web sites at a time [26, 27]. One of the problems with mechanisms employed at individual servers such as Apache's `mod_dosevasive` module [28] is that they are confined to local information and have hard-coded notions of what constitutes a denial-of-service attack. Consider a collection of a large number of compromised zombie machines that each send a steady rate of requests to a set of target victims. With sufficient interleaving, it may appear to each victim that each participant in the attack is sending only a trickle of requests. However, taken across all victims, a global view reveals malicious intent. The use of TCP puzzles as shown in Figure 5 elegantly mitigates coordinated attacks by forcing the source of the attack to perform work on the order of its global usage. Where local solutions are unable to identify and turn back such an attack, the ability for the set of victimized servers to push work back onto clients in times of heavy load can effectively mitigate such an attack, thus forcing a hacker to compromise a much larger number of servers in order to execute an effective attack against multiple sites.

4. RESEARCH ISSUES

The scenarios outlined above represent how network and transport puzzles would work in an ideal manner. There are an immense number of research issues, however, that must be addressed beyond simple protocol changes before the visions above can be realized. Several of the areas include:

4.1 Efficiency

To make puzzles resilient to denial of service attacks itself, the protocols and implementations of them need to be efficient. The ability to generate and give out puzzles and the ability, to validate puzzle answers should be orders of magnitude faster and cheaper than the ability to solve them. Specifically, it must be extremely hard for flooding attacks to saturate the system's ability to issue puzzles and it must be possible for the system to support a large number of puzzle answers using mechanisms such as protocol cookies [29]. Placing puzzles at the network layer exacerbates the problem as per-packet puzzles may be prohibitive on high-speed links. One approach for addressing this would be to leverage recent work in accounting and controlling high-bandwidth flows and high-bandwidth aggregates. Such mechanisms could be used to periodically trigger sufficiently difficult puzzles against the largest consumers of resources [30, 24, 31].

4.2 Tamper resistance

The puzzle mechanisms must not be susceptible to circumvention or misuse. In particular, the protocols and implementations need to guard against replay attacks and "answer" hijacking. It must also not be possible for malicious third parties to deny clients access to services by inserting themselves into or tampering with the puzzle protocols. Subversive attacks are especially important to guard against if puzzles are to be implemented at the IP layer. Consider a router that receives a puzzle from a downstream node. Due to the weakness of the Internet's identification mechanisms and the distributed nature of the routing infrastructure, the router is unable to determine the authenticity of the puzzle issuer's identity and whether or not the issuer is actually providing any service. An attacker either spoofing the IP

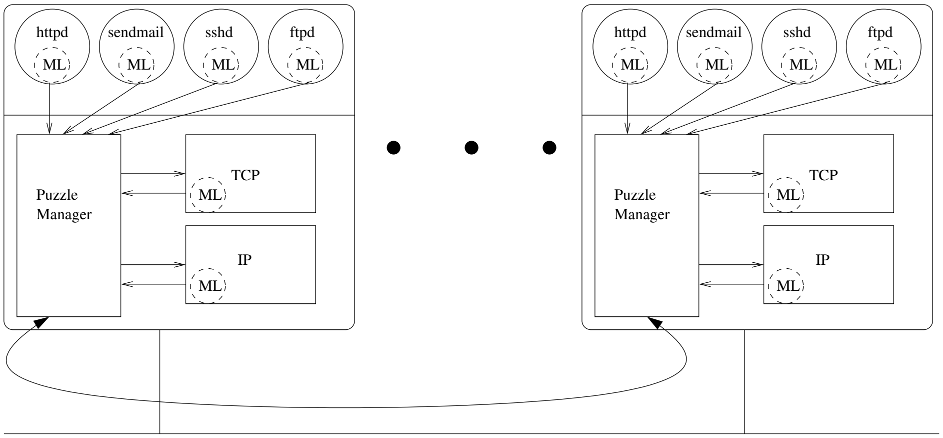


Figure 6: The Puzzle Manager

address of a legitimate source or claiming to be a legitimate source, must not be able disable the victim by falsely triggering the puzzle mechanisms against it. To support this, a tighter binding of identity to IP addresses would most likely be required along with more widespread support for accurately recording Internet paths. In general, we liken the deployment of puzzles to giving every network node a loaded gun to protect itself. While it is meant to do good things, it can be easily be used in inappropriate ways if not designed carefully.

4.3 Fairness

While puzzles can mitigate the effects of denial of service, without a puzzle generation algorithm that identifies and discriminates against known malicious activity, innocent victims can be impacted adversely. Part of developing an effective mechanism for thwarting denial of service attacks is to adaptively deliver harder puzzles to clients exhibiting suspicious behavior. For example, to mitigate magnification attacks, puzzle difficulty can be made proportional to the potential load being imparted on the system. An access router could simply pass back a puzzle 256 times harder than normal before allowing the forwarding of a packet that may be part of a Smurf or Fraggle attack on a class C network (i.e. an ICMP or UDP echo request to a broadcast IP address) [32]. Harder puzzles could also be given to those clients repetitively requesting the same content, those clients continuously requesting files that do not exist, and those clients attempting to exploit known vulnerabilities. Finally, while forcing malicious clients to perform puzzles before allowing access is important, the puzzle generation algorithm must also take into account “thin clients”, such as cell-phones or PDAs, and reduce the level of puzzle complexity to match the capabilities of end-point. To support such functionality, we believe an architecture that consists of a shared “Puzzle Manager” must be used so that a diverse number of applications across a diverse number of machines can appropriately identify and respond to threats. Similar to the Congestion Manager [33], the Puzzle Manager accepts input and recommendations from the network stack and from different applications on a particular machine as well as from other machines, to develop a more informed response. To generate appropriate signals, it is envisioned that each component would run some form of machine learning algorithm [34] driven by application-specific metrics (such as invalid requests) or by user feedback (such as spam feedback or recommender systems). Figure 6 outlines the architecture of the Puzzle Manager.

4.4 Control

Puzzles have clear similarity to explicit congestion notification (ECN) [35] and network congestion control in general. Puzzles are effectively multi-resolution, ECN signals that can not be ignored (assuming servers deny service to those not answering puzzles when given them). As with congestion control, control algorithms are essential for proper operation. There are several issues that make the control of puzzles much more difficult. The first is the fact that puzzles themselves can be adjusted to many levels. For a particular node, the network or end-host has the ability to dynamically determine the level of difficulty of the puzzle that is appropriate in order to control the node’s behavior. The second is the fact that the same puzzle has non-uniform impact across different nodes. This is in direct contrast to TCP with ECN, which halves the rate of the sender upon notification. Given these issues, control algorithms that work well similar to the work that has already been done with TCP [36, 37, 38, 39] and active queue management (AQM) [40, 41, 42, 43, 44] must be developed with the overall goal of maximizing resource utilization while providing equitable fairness amongst clients [45]. Another aspect in puzzle control, would be to use the mechanisms to support some forms of QoS. For example, based on the difficulty of the puzzle, its solution could be used as “credit” towards better service. Given this, a “Puzzle Fair Queuing” (PFQ?) algorithm could be designed to weight service based on the number of credits accumulated by individual flows and flow aggregates.

4.5 Deployment

Without a proper financial incentive to deploy puzzles, such ideas will go the way of QoS over the Internet [46]. Unlike QoS, however, there are much clearer monetary benefits for thwarting DDoS attacks and eliminating the downtime associated with them. On the server side, with the large estimated costs of the SQL Slammer worm ($1.2 billion), the Code Red worm ($2.6 billion), the LoveLetter virus ($8.8 billion), and the Klez virus ($9.0 billion) [47], individual service providers and businesses have enormous potential financial benefit. In addition, such a mechanism could even be used as another means for slowing down spam, whose cost in terms of lost productivity has not been quantified. On the client-side, with widespread server support for pushing puzzles back to clients during overload situations, there is clear incentive for clients to support puzzles in order to maintain some level of service in times of heavy load induced by DDoS attacks.

5. CONCLUSION

Network and transport level puzzles have the potential to significantly change the ability to launch distributed denial of service attacks. However, it is clearly an open research question as to whether or not protocols and systems can be designed and built to achieve the goal of minimizing the impact of such attacks. Issues such as efficiently generating puzzles, hardening puzzles against subversion, controlling puzzles to maximize server utilization, adapting puzzles to achieve equitable fairness, and creating incentives for deployment must all be addressed. As part of an ongoing project in TCP/IP puzzles [48], we are currently implementing the necessary protocol changes and developing initial solutions to the issues described above. We plan on releasing this code to provide a platform for researchers to further explore the issues involved in protecting the Internet via TCP/IP puzzles.

6. REFERENCES

[1] D. Clark, “NSF ANIR PI Meeting Keynote Address,” Reston, VA, 2003.

[2] D. Moore, C. Shannon, and J. Brown, “Code-Red: A Case Study on the Spread and Victims of an Internet Worm,” in *Internet Measurement Workshop*, November 2002.

[3] L. Wang, X. Zhao, D. Pei, R. Bush, D. Massey, A. Mankin, S. Wu, and L. Zhang, “Observation and Analysis of BGP Behavior Under Stress,” in *Internet Measurement Workshop*, November 2002.

[4] J. Jung, B. Krishnamurthy, and M. Rabinovich, “Flash Crowds and Denial of Service Attacks: Characterization and Implications for CDNs and Web Sites,” in *International World Wide Web Conference*, May 2002, pp. 252–262.

[5] F. Kargl, J. Maier, and M. Weber, “Protecting Web Servers from Distributed Denial of Service Attacks,” in *World Wide Web*, 2001, pp. 514–524.

[6] S. Staniford, V. Paxson, and N. Weaver, “How to 0wn the Internet in Your Spare Time,” in *11th USENIX Security Symposium (Security ’02)*, 2002.

[7] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver, “The Spread of the Sapphire/Slammer Worm,” http://www.caida.org/outreach/papers/2003/sapphire/, 2003.

[8] A. Juels and J. Brainard, “Client Puzzles: A Cryptographic Defense Against Connection Depletion,” in *NDSS*, 1999, pp. 151–165.

[9] D. Dean and A. Stubblefield, “Using Client Puzzles to Protect TLS,” in *10th Annual USENIX Security Symposium*, 2001.

[10] T. Aura, P. Nikander, and J. Leiwo, “DOS-Resistant Authentication with Client Puzzles,” *Lecture Notes in Computer Science*, vol. 2133, 2001.

[11] J. Leiwo, T. Aura, and P. Nikander, “Towards Network Denial of Service Resistant Protocols,” in *SEC*, 2000, pp. 301–310.

[12] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. Wallach, “Security for Peer-to-Peer Routing Overlays,” in *Proceedings of OSDI*, December 2002.

[13] M. Abadi, M. Burrows, M. Manasse, and T. Wobber, “Moderately Hard, Memory-bound Functions,” 2003.

[14] I. Clarke, O. Sandberg, B. Wiley, and T. Hong, “Freenet: A Distributed Anonymous Information Storage and Retrieval System,” *Lecture Notes in Computer Science*, vol. 2009, pp. 46+, 2001.

[15] B. Braden, N. Chiappa, D. Clark, T. Faber, A. Falk, M. Handley, S. Shenker, K. Sollins, and J. Wroclawski, “NewArch Project: Future-Generation Internet Architecture,” http://www.isi.edu/newarch/, 2003.

[16] J. Postel, D. Johnson, T. Markson, B. Simpson, and Z. Su, “ICMP Type Numbers,” http://www.iana.org/assignments/icmp-parameters, 2001.

[17] B. Carpenter, D. Estrin, D. Farinacci, G. Finn, C. Graff, D. Katz, J. Postel, A. Malis, and R. Ullman, “IP Option Numbers,” http://www.iana.org/assignments/ip-parameters, 2001.

[18] S. Bellovin, B. Braden, M. Bridges, S. Knowles, J. Kay, K. Scott, S. Subramaniam, and V. Sukonnik, “TCP Option Numbers,” http://www.iana.org/assignments/tcp-parameters, 2001.

[19] Fyodor, “Remote OS detection via TCP/IP Stack FingerPrinting,” http://www.insecure.org/nmap/, 1998.

[20] R. Deraison, “Nessus,” http://www.nessus.org/, 2003.

[21] D. Kaminsky, “Doxpara: Paketto Keiretsu (scanrand),” http://www.doxpara.com/read.php/code/paketto.html, 2002.

[22] E. Spafford, “The Internet Worm: Crisis and Aftermath,” *Communications of the ACM*, vol. 32, no. 6, pp. 678–698, June 1989.

[23] W. Cheswick and S. Bellovin, “Firewalls and Internet Security,” .

[24] R. Mahajan, S. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker, “Controlling High

Bandwidth Aggregates in the Network," *Computer Communication Review*, vol. 32, no. 3, July 2002.

[25] D. Kaminsky, "Black Ops of TCP/IP: Paketto Keiretsu 1.0 Release," `http://slashdot.org/article.pl?sid=02/11/18/2032225&mode=thread`, 2003.

[26] U.S. Department of Justice: Federal Bureau of Investigation, "Mafiaboy," `http://www.fbi.gov/pressrel/pressrel00/mafia080700.htm`, 2000.

[27] V. Paxson, "An Analysis of Using Reflectors for Distributed Denial-of-Service Attacks," *Computer Communication Review*, vol. 31, no. 3, July 2001.

[28] Network Dweebs Corporation, "Apache DoS Evasive Maneuvers Module," `http://www.networkdweebs.com/stuff/security.html`, 2003.

[29] D. Bernstein, "SYN Cookies," `http://cr.yp.to/syncookies.html`, 2003.

[30] C. Estan and G. Varghese, "New Directions in Traffic Measurement and Accounting," in *Internet Measurement Workshop*, November 2001, pp. 75–80.

[31] W. Feng, D. Kandlur, D. Saha, and K. Shin, "Stochastic Fair Blue: A Queue Management Algorithm for Enforcing Fairness," in *Proc. of INFOCOM*, April 2001.

[32] C. Huegen, "Network-based Denial of Service Attacks," http://www.nanog.org/mtg-9806, June 1998.

[33] H. Balakrishnan and S. Seshan, "The Congestion Manager," *RFC 3124*, June 2001.

[34] T. Mitchell, *Machine Learning*, McGraw Hill, 1997.

[35] K. Ramakrishnan and S. Floyd, "A Proposal to Add Explicit Congestion Notification (ECN) to IP," *RFC 2481*, January 1999.

[36] L. Brakmo, S. O'Malley, and L. Peterson, "TCP Vegas: New Techniques for Congestion Detection and Avoidance," in *Proceedings of ACM SIGCOMM*, October 1994, pp. 24–35.

[37] V. Jacobson, "Modified TCP Congestion Avoidance Algorithm," *end2end-interest mailing list (ftp://ftp.ee.lbl.gov/email/vanj.90apr30.txt)*, April 1990.

[38] W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms," *RFC 2001*, January 1997.

[39] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP Throughput: A Simple Model and its Empirical Validation," in *Proceedings of ACM SIGCOMM*, September 1998.

[40] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *ACM/IEEE Transactions on Networking*, vol. 1, no. 4, pp. 397–413, August 1993.

[41] C. Hollot, V. Misra, D. Towsley, and W. Gong, "On Designing Improved Controllers for AQM Routers Supporting TCP Flows," in *INFOCOM*, 2001, pp. 1726–1734.

[42] T. Ott, T. Lakshman, and L. Gong, "SRED: Stabilized RED," in *Proceedings of IEEE INFOCOM*, March 1999.

[43] S. Athuraliya, S. Low, V. Li, and Q. Yin, "REM: Active Queue Management," *IEEE Network Magazine*, vol. 15, no. 3, May 2001.

[44] W. Feng, D. Kandlur, D. Saha, and K. Shin, "The Blue Queue Management Algorithms," *IEEE/ACM Transactions on Networking*, vol. 10, no. 4, August 2002.

[45] D. Mankins, R. Krishnan, C. Boyd, J. Zaho, and M. Frentz, "Mitigating Distributed Denial of Service Attacks with Dynamic Resource Pricing," in *Proceedings of Annual Computer Security Applications Conference (ACSAC 2001)*, 2001.

[46] ACM SIGCOMM, "`http://www.acm.org/sigcomm/sigcomm2003/workshop/ripqos/`," Workshop on Revisiting IP QoS: Why Do We Care, What Have We Learned? (RIPQOS), 2003.

[47] R. Lemos, "Counting the Cost of Slammer," `http://news.com.com/2100-1001-982955.html`, January 2003.

[48] W. Feng, "PuzzleNet Project," `http://www.cse.ogi.edu/sysl/projects/puzzles`.