Editor: Siobhán Clarke • siobhan.clarke@cs.tcd.ie

Denial-of-Service Attack-Detection Techniques

Glenn Carl and George Kesidis • *Pennsylvania State University*
Richard R. Brooks • *Clemson University*
Suresh Rai • *Louisiana State University*

Denial-of-service (DoS) detection techniques — such as activity profiling, change-point detection, and wavelet-based signal analysis — face the considerable challenge of discriminating network-based flooding attacks from sudden increases in legitimate activity or flash events. This survey of techniques and testing results provides insight into our ability to successfully identify DoS flooding attacks. Although each detector shows promise in limited testing, none completely solve the detection problem. Combining various approaches with experienced network operators will most likely produce the best results.

The Internet was designed for the minimal processing and best-effort forwarding of any packet, malicious or not. For cyberattackers – motivated by revenge, prestige, politics, or money – this architecture provides an unregulated network path to victims. Denial-of-service (DoS) attacks exploit this to target mission-critical services. A quantitative estimate of worldwide DoS attack frequency found 12,000 attacks over a three-week period in 2001.[1] The 2004 *CSI/FBI Computer Crime and Security Survey*[2] listed DoS attacks among the most financially expensive security incidents. The magnitude of the incidence rate and potential recovery expense has garnered the interest of security managers and researchers alike.

DoS attacks, which come in many forms, are explicit attempts to block legitimate users' system access by reducing system availability. We could, for example, consider the intentional removal of a system's electrical power as a physical DoS attack. An attacker could also render a computing resource unavailable by modifying the system configuration (such as its static routing tables or password files). Such physical or host-based intrusions are generally addressed through hardened security policies and authentication mechanisms. Although software patching defends against some attacks, it fails to safeguard against DoS flooding attacks, which exploit the unregulated forwarding of Internet packets. A secondary defense that includes both attack detection and countermeasures is required.

Here, we survey various approaches for detecting DoS flooding attacks – a *network-based* attack in which agents intentionally saturate system resources with increased network traffic. In a distributed DoS (DDoS) attack, the assault is coordinated across many hijacked systems (*zombies*) by a single attacker (*master*). Techniques that detect DoS also apply to DDoS. (We don't discuss defense or countermeasures; these are surveyed elsewhere,[3] and typically include using packet filters to stem the attack's packet flow.) The malicious workload in network-based DoS attacks comprises network datagrams or packets that consume network buffers, CPU processing cycles, and link bandwidth. When any of these resources form a bottleneck, system performance degrades or stops, impeding legitimate system use. Overloading a Web server with spurious requests, for example, slows its response to legitimate users. This specific DoS attack type doesn't breach the end (victim) sys-

1089-7801/06/$20.00 © 2006 IEEE

tem, either physically or administratively, and requires no other pre-existing conditions except an Internet connection.

Network-Based DoS Attacks

Although many high-profile DoS attacks have occurred, few have been empirically captured and analyzed. Given the potential for bad publicity, victims hesitate to share information regarding security incidents. As a result, it's difficult for researchers to directly observe attacks and find their ubiquitous characteristics. In cases in which attack forensics are available, researchers can introduce classification systems, but attackers typically modify their techniques soon after discovery. As a result, the DoS attack-definition space is ever changing.

General Attack Types

To keep our discussion manageable, we've generalized it based on the exploited weakness, dividing the network-based DoS attack space into vulnerability attacks and flooding attacks. A more detailed classification of DoS attacks is available elsewhere.[4]

In a *vulnerability* attack, malformed packets interact with some network protocol or application weakness present at the victim. This type of vulnerability typically originates in inadequate software assurance testing or negligent patching. The malformed attack packets interact with installed software, causing excessive memory consumption, extra CPU processing, system reboot, or general system slowing. Popular examples are the land attack, Neptune or Transmission Control Protocol synchronization (TCP SYN) flag, the ping o' death, and the targa3 attacks.

Flooding attacks — our focus here — send the victim a large, occasionally continuous, amount of network traffic workload. As a result, legitimate workloads can become congested and lost at bottleneck locations near or removed from the victim. Such an attack requires no software vulnerability or other specific conditions. To saturate network links, queues, and processors with workload anywhere in the network, the attack can use a range of protocols, including Internet Control Message Protocol (ICMP), User Datagram Protocol (UDP), and TCP, through tools such as stream2, synhose, synk7, synsend, and hping2. Under continued attack-related congestion, flow-controlled applications will continue to increase their back-off time between retransmissions. From the users' perspective, their workload isn't being processed; a DoS situation has occurred.

Attack Detection

Vulnerability-attack workloads use common attributes to exploit software weaknesses. A TCP SYN attack, for example, requires repetitive use of specific TCP flag fields. Once the exploit is identified, adequate vendor support ensures the vulnerability is short-lived and unlikely to return. Vendors can address TCP SYN attacks using `syn cache`, `syn cookies`, and `synkill` mechanisms, for example.

Although vendors can address vulnerability attacks by correcting protocol or application weaknesses, these types of attacks can remain problematic. If their volume is sufficient enough to cause resource depletion and subsequent performance degradation, they can be reclassified as flooding attacks. For this reason, flooding attacks are especially difficult because even the best-maintained system can become congested, thus denying service to legitimate users.

Survey of Detection Approaches

A detector's main goal is to detect and distinguish malicious packet traffic from legitimate packet traffic. If, for example, many clients all want Web service and a DoS attack maliciously floods many Web session requests as well, how can the Web server discriminate between the requests? Clearly, legitimate user activity can be easily confused with a flooding attack, and vice versa.

When large amounts of expected or unexpected traffic from legitimate clients suddenly arrive at a system, it's called a *flash event*. One way to predict such events and thus distinguish them from DoS attacks is for service providers to be aware, a priori, that adding new content might trigger large request volume.[5] Unpredictable and legitimate Web activity is also possible, however (as with the Slashdot effect, in which a newly posted link on a popular news or information site results in numerous Web requests). Because there is no innate Internet mechanism for performing malicious traffic discrimination, our best alternative is to install attack detectors to monitor real-time traffic, rather than rely on static traffic load predictions.

DoS attack-detection approaches can be installed locally, thus protecting a possible victim, or remotely, to detect propagating attacks. Although detecting propagating attacks is desirable, IT departments generally focus on protecting their own networks and therefore choose local detection approaches. In this case, they place detectors at the potential victim resource or at a router or firewall within the victim's subnetwork. Under this assumption, we have limited our scope to that of the victim, which excludes several other potential detection methods, such as the source-based DWARD[6], traceback, path identification, and others.

A survey of detection methods reveals disparate uses of test data, different attack types, and a wide range of reported results.

All detection methods define an attack as an abnormal and noticeable deviation of some statistic of the monitored network traffic workload. Clearly, the choice of statistic is critically important. Each of the following groupings of attack detection techniques includes an evaluation of a different statistic of network traffic.

Activity Profiling

Monitoring a network packet's header information offers an *activity profile*. Loosely defined, this activity profile is the average packet rate for a network flow, which consists of consecutive packets with similar packet fields (such as address, port, and protocol). The elapsed time between consecutive matching packets determines the flow's average packet rate or activity level. We can measure total network activity as the sum over the average packet rates of all inbound and outbound flows.

To analyze individual flows for all possible UDP services, we would have to monitor on the order of 2^{64} flows, and including other protocols, such as TCP, ICMP, and Simple Network Management Protocol (SNMP) greatly compounds the number of possible flows. To avoid high-dimensionality issues, we can *cluster* individual flows with similar characteristics. Each cluster's activity level is the summation of constituent flows. For this abstraction, an attack is indicated by

- increasing activity levels among clusters, which can indicate a few attacking agents increasing their attack-generation rate; or
- an increase in the overall number of distinct clusters, which can represent many distributed attacking agents (as in a DDoS).

In the backscatter analysis project,[1] researchers monitored a wide IP address space for incoming unsolicited "backscatter" packets. Such packets are a non-collocated victim's response to several spoofed vulnerability and flooding attacks. The backscatter packets' source address is that of the victim, but the packet's destination address is randomly spoofed. An attack that uses uniformly distributed address-spoofing leads to a finite probability that any monitored address space will receive backscatter packets. At the monitoring point, captured backscatter packets are clustered based on the unique victim source address. To detect attacks, the researchers analyze a cluster's destination address distribution uniformity using an Anderson-Darling test statistic, in addition to thresholding the cluster's activity level (the attack rate) and lifetime.

Laura Feinstein and her colleagues focus their detection efforts on activity level and source address distribution.[7] They cluster flows according to the addresses of the destination machines located behind the monitoring point. The first cluster contains the single most frequently seen source address, the second cluster contains the next four most frequent, the third cluster the next 16, the fourth the next 256, and the fifth the next 4,096; the sixth cluster encompasses all remaining traffic. The researchers compare each cluster's activity level to the expected amount using a chi-square statistic, thus providing a "goodness of fit" result. A deviation from the expected traffic profile suggests anomalous activity, and is detectable by thresholding the chi-square statistic's magnitude.

Many other address-distribution statistics are possible, including entropy, which is considered a measure of randomness. Attacks that use uniform address distributions will maximize the entropy statistic, whereas one large voluminous flow will minimize the entropy. Thresholding an entropy deviation from the expected traffic's source address profile can suggest anomalous activity.[7]

Sequential Change-Point Detection

Change-point detection algorithms isolate a traffic statistic's change caused by attacks. These approaches initially filter the target traffic data by address, port, or protocol and store the resultant flow as a time series. The time series can be considered a time-domain representation of a cluster's activity. If a DoS flooding attack begins at time λ, the time series will show a statistical change either around or at a time greater than λ.

One class of change-point detection algorithms operates on continuously sampled data and requires only low amounts of memory and computational resources. An example here is cumulative sum (Cusum) algorithms. To identify and localize a DoS attack, the Cusum identifies deviations in the actual versus expected local average in the traffic time series.[8–10] If the difference exceeds some upper bound, the Cusum's recursive statistic increases for each time-series sample. During time intervals containing only normal traffic, the difference is below this bound, and the Cusum statistic decreases until reaching zero. Using an appropriate threshold against the Cusum statistic, the algorithm identifies an increasing trend in the time-series data, which might indicate a DoS attack's onset. Through the settings of the threshold and upper bound, the Cusum algorithm can trade off detection delay and false-alarm rates. Other researchers have extended this detection method to identify the typical scanning activities of network worms.[11]

Wavelet Analysis

Wavelet analysis describes an input signal in terms of spectral components. Although Fourier analysis is more common, it provides a global frequency description and no time localization. Wavelets provide for concurrent time and frequency description, and can thus determine the time at which certain frequency components are present. For detection applications, wavelets separate out time-localized anomalous signals from background noise; the input signal contains both. Ideally, the signal and noise components will dominate in separate spectral windows. Analyzing each spectral window's energy determines the presence of anomalies.

Paul Barford and his colleagues[12] define anomalies as network failures or misconfigurations, attacks (DoS or other), flash events, and other “measurement” events. They decomposed traffic data into distinct time series of average IP/HTTP packet sizes per second, flows per second, and bytes per second. They then applied wavelet analysis to each time series, resulting in time-localized high- and mid-band spectral energies. They considered low-frequency content to be daily or weekly activity, and thus not an onset of an abrupt attack. To identify anomalies, they weighted a combination of high- and middle-spectral energies, and then thresholded its variability.

Wavelet energies in the high-band spectral window can also identify change points within an input signal. To enhance a Cusum change-point detection approach's performance, Richard Brooks and his colleagues used discrete wavelet analysis to postprocess the Cusum statistic's response.[10] The signed magnitude of the high-band wavelet energy is proportional to the abruptness of an increasing Cusum statistic. Thresholding the high-band spectral energies quantifies the Cusum's abruptness, which is a potential indicator of an abrupt flooding attack.

Detection Method Results

Surveying each detection method's validation reveals disparate uses of test data, different attack types, and a wide range of reported results. In most cases, researchers provided quantitative true detection results, but didn't provide false positives, missed detections, and detection delay results. Table 1 summarizes the testing conditions and noteworthy detection test results.

Backscatter Analysis

Researchers[1] analyzed the backscatter within three weeks' worth of empirical data from an ingress link supporting 2^{24} IP addresses. Conservative results indicated that more than 12,000 DoS attacks were attempted, involving 5,000 distinct victims' IP addresses. The researchers suggested that 50 percent of those attacks were either TCP SYN floods or closed port probes, and 15 percent were ICMP responses from TCP floods. Overall, 90 percent of the attacks used TCP ranging across various services, including Internet Relay Chat (IRC), HTTP, Telnet , and Authd. Almost half the attacks (46 percent) had an estimated rate of 500+ packets per second.

Chi-Square/Entropy Detector

Researchers tested the chi-square and entropy detector[7] against a small set of six publicly available data sets with anonymized IP addresses. The networking environments included a peering Internet service provider (NZIX), a 450-person research organization (Bell Labs), a small university (Ohio University), and a small company. The total amount of data appeared to be between 100 and 150 hours, with data rates ranging from 1 to 16 Mbits per second. Because the data traces included no known

Table 1. Testing Summary

Detection method	Reference	Test data	Attack description	False-positive rate	Detection delay	Detection results	Memory (1 = lowest)	Complexity (1 = lowest)
Activity profiling	1	Three weeks' worth of private network data	"Backscatter" response packets from TCP SYN, TCP flood, and closed port probes	—	—	12,000 DoS attacks on 5,000 distinct victims	6	6
	7	Six publicly available data sets	Stacheldraht ICMP, TCP SYN, and UDP flood attack overlay of 25 percent intensity; victims' addresses randomly chosen from a uniform distribution	—	—	2 out of 2 attacks detected	3	3
Change-point detection	8	ns-2 simulation of 100 nodes	TCP, UDP, and ICMP floods by abrupt and linear increase	1–6 alarms per 100 time-series samples	1–36 seconds	UDP abrupt/ linear flood ICMP abrupt/ linear flood	1	1
	9	Three private network data sets	TCP SYN constant rate flood attack	—	20 seconds to 8 minutes	100% detection with rate of >35 SYNs per second; 70% detection at 33 SYNS per second	1	1
Wavelet analysis	12	Three weeks' worth of university data	119 DoS abrupt flood attacks of 4x, 7x, and 10x intensities overlaid on empirical data	21% false detection rate over 238 time series	Average: 25 seconds	47% detection rate over 119 time series	4	4
	10	Three weeks' worth of university data with 109 anomalies	39 recorded anomalies, including some DoS floods	—	5 minutes to 1.5 hours	38 out 39 anomalies	5	5

DoS attacks, the researchers added overlaid attack traffic provided by the Stacheldraht DDoS attack tool. Stacheldraht – which means "barbed wire" in German – performs ICMP, SYN, and UDP floods that can run for a specified duration.

The study's first test experiment overlaid the public data set with 25 percent attack packets. A second experiment removed 25 percent of the traffic and replaced it with attack packets. In both cases, the attack packets' source addresses were drawn from a uniform distribution. The entropy and chi-square detector provided positive attack indication for both test cases.

Cusum and Wavelet Approaches

To test the Cusum sequential change-point detection against UDP, TCP, and ICMP traffic floods, researchers used the ns-2 simulator to construct a network of 100 nodes.[8] Of those nodes, four were core transit nodes and the remaining 96 nodes were distributed into 12 edge domains. Background traffic was a mixture of ICMP, UDP, and TCP protocols, with TCP accounting for more than 75 percent of the traffic.

The researchers performed three attack simulations: TCP SYN, UDP, and ICMP floods. Each attack reached 20 percent of the total aggregate traffic through either linear or abrupt increases. Cusum detected most of the attacks. In addition, the researchers confirmed the theoretical and experimental relationships between detection delay and false-alarm rate. False-alarm rates ranged from less than 1 to 6 alarms per 100 packets monitored. Researchers observed detection delays ranging from 1 to 36 seconds, depending on desired false-alarm rate.

Another study used a Cusum algorithm against TCP SYN attacks.[9] The three test data set sources included a large company's wide-area Internet access point (10 Mbits per second) and two university's Internet access points (10 and 622 Mbits per second). From the test data, the researchers extracted TCP traffic containing SYN, ACK (acknowledge), and RST (reset) flags into a time series, and then overlaid it with TCP SYN floods of constant intensity. They used rates of 33 to 100 TCP SYNs per second. For attacks above 33 and 35 SYNs per second, Cusum's detection probability was 70 and 100 percent, respectively. Detection delays ranged from 20 seconds to 8 minutes.

Using wavelet analysis, researchers evaluated six months' worth of router SNMP and IP flow records, sampled at 5-minute intervals.[12] The monitoring point was a university-based wide-area access point. Network engineering analysis of the log data identified more than 100 anomalous events, including network malfunctions, attacks, and flash events. The researchers used a subset of 39 high-confidence anomalies for detection evaluation, although it's unclear how many of the anomalies were specifically DoS attacks. The wavelet analysis missed only one of the anomalies. Detection time had an ambiguity of 1.5 hours.

In another study,[10] researchers used both Cusum and a wavelet detector to analyze three weeks' worth of empirical data collected from a university gateway. The researchers separately superimposed 119 DoS flooding attacks on this data at intensities of four, seven, and 10 times that of the background rate. Using a Cusum detection algorithm against the 4x DoS attack, they measured 15 percent true detection and 18 percent false positives for a detection rate of 0.83. Adding wavelet processing raised these metrics to 40 percent true detections and 30 percent false positives for a detection rate of 1.3. Wavelet processing provided a 56 percent increase in detection efficiency over the Cusum alone. Using parameter tuning can slightly improve the wavelet's true detection rates to above 47 percent, with a decline in the false-positive rate to 21 percent.

Outstanding Concerns

DoS detectors aim to differentiate legitimate from malicious traffic under a wide range of operating conditions. Although the surveyed detectors do indeed detect some examples of DoS attacks, core problems are apparent.

Varying Test Conditions

Most detectors we surveyed were woefully undertested against varying network and attack conditions. Comprehensive testing is obviously a highly complex, time-consuming process, calling for more efficient and comprehensive approaches. Existing studies employed little variation in network environment, attack-rate dynamics, or address spoofing to emulate a realistic deployment setting. Researchers must include flash events and other legitimate activities that closely mimic attack activity in all test traffic. Of the surveyed detectors, only one explicitly acknowledged the presence of flash events.

This undertesting problem is partly due to the unavailability of comprehensive test data, testing environments, and standards. We hope that such issues will be addressed by upcoming cybersecurity initiatives, including the Cyber Defense Technology Experimental Research Project (Deter; www.isi.edu/deter/docs/testbed.overview.htm) and the Protected Repository for the Defense of Infrastructure against Cyber Threats (Predict; www.hsarpacyber.com/ongoing.html#datasets).

Measuring Network Activity

Attack-detection statistics are only relative to "normal" network activity. Attack models with sharp volume increases or uniform address distributions reflect a small, aging subset of the attack problem space. These are properties of earlier generational DoS attack tools, and as is well known, attackers change their tools soon after discovery.

In all detector schemes, researchers have yet to develop nominal-traffic measures that encompass the range of possible networks conditions. To quantify normal activity, we must know the expected activity level of services running on the network's various machines. We can estimate network service activity using information provided by network administrators, port probing, or direct traffic monitoring. Yet, normal activity is a varying process: network services have different lifetimes, activity levels, and availability, in keeping with users' variable time-of-day interactions. At this point, it's unclear whether suitable training algorithms or rule-of-thumb guidelines exist that can adequately model nominal traffic's irregular behavior.

Traffic-flow clustering offers insight into network activity, and trained network analysts can easily visualize it with some of today's tools. Quantitative measures, such as wavelet energies, lack this desirable property. Nonetheless, a strong dependency naturally exists between clustering and detection. Defining a cluster is inherently complex for a given network and can be difficult to validate. We've yet to see real-time, truly scalable algorithms that can create, destroy, and modify the clusters with no a priori application or protocol knowledge.

An 'attack' is an abnormal and noticeable deviation of some statistic of the monitored network traffic workload.

Subverting Detection

Most researchers concede that attackers can defeat their detection methods by developing attacks through trial and error. When studying activity profiling or evaluating entropy of an address distribution, researchers assume that the regular traffic is distributed among a few clusters or flows. It seems reasonable that attackers can sniff local traffic, understand the address distribution, and then spoof the addresses based on this calculation. For change-point detectors that monitor changes in packet volume over time, an initially low, slowly ramping attack rate dynamic might be obscured by the background traffic's high variability.

Setting Parameters

Each detector has multiple operating parameters, including clustering configuration, sampling window size, thresholds, and wavelet filtering level. In most cases, researchers offer no guidance on parameter variations or their effect on performance. Indeed, researchers often optimize parameters to their own experimental test cases. When it comes to deployment, users have few clues as to how they might adjust the detection performance for their own environments; ad hoc training in parameter settings is typically required.

Implementation Issues

None of the studies we reviewed addressed real-world implementation concerns. In Table 1, we offer our relative rankings of the detection methods' computational complexity and memory use. Because the Cusum algorithms[8,9] are based on single-stage, recursive, exponentially weighted estimators, they're the least complex and have the lowest memory use. The chi-square/entropy detector[7] is next because it uses only six bins to analyze its address distributions. More detailed clustering, however, will increase its complexity and memory requirements.

The combined Cusum and wavelet-based method[10] incurs an extra $O(2^n)$ complexity over the Cusum methods, where $5 < n < 11$ is the spectral-resolution level. Compared to this method, Barford's wavelet method[12] is at least two times as complex because it uses two redundant wavelet filter stages. David Moore and his colleagues[1] analyze the largest address distribution — a /8 network (2^{24} individual addresses) — and their method consequently requires the most computation and memory use.

For network administrators, security is a fundamental concern, and they must have efficient, reliable tools to help them quickly recognize and investigate anomalous activities. Although intrusion detection is immature and doesn't always detect malicious activity, it can provide administrators with a useful diagnostic resource.

At this point, we've yet to find a single technique to adequately detect DoS flooding attacks; combining approaches might offer the best performance.[13,14] Because false positives are likely in any case, however, experienced network administrators are crucial in the attack-identification effort.

Acknowledgments

Our work was supported by a US National Science Foundation grant (CCR0310916) and the Pennsylvania State University Applied Research Laboratory's Educational and Foundational program.

References

1. D. Moore, G.M. Voelker, and S. Savage, "Inferring Internet Denial-of-Service Activity," *Proc. Usenix Security Symp.*, Usenix Assoc., 2001; http://citeseer.ist.psu.edu/moore01 inferring.html.
2. L. Gordon et al., *CSI/FBI Computer Crime and Security Survey*, Computer Security Inst., 2004; http://i.cmpnet.com/gocsi/db_area/pdfs/fbi/FBI2004.pdf.
3. J. Mirkovic et al., *Internet Denial of Service: Attack and Defense Mechanisms*, Prentice Hall, 2005.
4. J. Mirkovic, J. Martin, and P. Reiher, "A Taxonomy of DDoS Attacks and DDoS Defense Mechanisms," *ACM Sigcomm Computer Comm. Rev.*, vol. 34, no. 2, 2004, pp. 39–53.
5. J. Jung, B. Krishnamurthy, and M. Rabinovich, "Flash Crowds and Denial of Service Attacks: Characterization and

Implications for CDNs and Web Sites," *Proc. Int'l World Wide Web Conference*, ACM Press, 2002, pp. 252–262.

6. J. Mirkovic, G. Prier, and P. Reiher, "Attacking DDoS at the Source," *Proc. 10th Int'l Conf. Network Protocols* (ICNP 2002), IEEE CS Press, 2002, pp. 312–321.
7. L Feinstein et al., "Statistical Approaches to DDoS Attack Detection and Response," *Proc. DARPA Information Survivability Conf. and Exposition*, vol. 1, 2003, IEEE CS Press, pp. 303–314.
8. R.B. Blazek et al., "A Novel Approach to Detection of 'Denial-of-Service' Attacks via Adaptive Sequential and Batch-Sequential Change-Point Detection Methods," *Proc. IEEE Workshop Information Assurance and Security*, IEEE CS Press, 2001, pp. 220–226.
9. H. Wang, D. Zhang, and K. Shin, "Detecting SYN Flooding Attacks," *Proc. 21st Joint Conf. IEEE Computer and Comm. Societies (IEEE INFOCOM)*, IEEE Press, 2002, pp. 1530–1539.
10. R.R. Brooks, *Disruptive Security Technologies with Mobile Code and Peer-to-Peer Networks*, CRC Press, 2005.
11. J. Jung et al., "Fast Portscan Detection Using Sequential Hypothesis Testing," *Proc. IEEE Symp. Security and Privacy*, IEEE CS Press, 2004, pp. 211–225.
12. P. Barford et al., "A Signal Analysis of Network Traffic Anomalies," *Proc. ACM SIGCOMM Internet Measurement Workshop*, ACM Press, 2002, pp. 71–82.
13. J. Allen et al., *State of the Practice of Intrusion Detection Technologies*, tech. report CMU/SEI-99-TR-028, Software Eng. Inst., Carnegie Mellon Univ., 2000.
14. R. Lippmann et al., "The 1999 DARPA Off-Line Intrusion Detection Evaluation," *Computer Networks*, vol. 34, no. 4, 2000, pp. 579–595.

Glenn Carl is a Phd student in electrical engineering at Pennsylvania State University. His research interests include intrusion and anomaly detection, network security testing, and large-scale network simulation. Contact him at gmc102@psu.edu.

Richard R. Brooks is an associate professor in the Holcombe Department of Electrical and Computer Engineering at Clemson University. His research interests are in strategic distributed systems, network security, and adaptive infrastructure. Brooks has a BA in mathematical sciences from the Johns Hopkins University and a PhD in computer science from Louisiana State University. Contact him at rrb@acm.org.

Suresh Rai is a professor in the Electrical and Computer Engineering Department at Louisiana State University. His research interests include network traffic, wavelet-based compression, and security. Rai has a PhD from Kurukshetra University, India. Contact him at suresh@ece.lsu.edu.

George Kesidis is an associate professor in Pennsylvania State University's computer science and engineering and electrical engineering departments. His current interests are in routing for wireless mobile ad hoc networks, network pricing and economics, and cybersecurity testing. Kesidis has a PhD in electrical engineering and computer science from the University of California, Berkeley. Contact him at kesidis@engr.psu.edu.

Write for Spotlight

Spotlight focuses on emerging technologies, or new aspects of existing technologies, that will provide the software platforms for Internet applications.

Spotlight articles describe technologies from the perspective of a developer of advanced Web-based applications. Articles should be 2,000 to 3,000 words. Guidelines are at www.computer.org/internet/dept.htm.

To check on a submission's relevance, please contact department editor Siobhán Clarke at siobhan.clarke@cs.tcd.ie.