Also in: Proceedings of IEEE WETICE 2003 Workshop on Enterprise Security, Linz, Austria, June 2003

Indra: A peer-to-peer approach to network intrusion detection and prevention

Ramaprabhu Janakiraman
Department of Computer Science and Engineering
Washington University in St. Louis
rama@arl.wustl.edu

Marcel Waldvogel
IBM Research
Zurich Research Laboratory
mwl@zurich.ibm.com

Qi Zhang
Microsoft Inc.
qz@cs.wustl.edu

***Abstract*— While the spread of the Internet has made the network ubiquitous, it has also rendered networked systems vulnerable to malicious attacks orchestrated from anywhere. These attacks or intrusions typically start with attackers infiltrating a network through a vulnerable host and then launching further attacks on the local network or Intranet. Attackers rely on increasingly sophisticated techniques like using distributed attack sources and obfuscating their network addresses. On the other hand, software that guards against them remains rooted in traditional centralized techniques, presenting an easily-targeted single point of failure. Scalable, distributed network intrusion prevention techniques are sorely needed.**

We propose Indra—a distributed scheme based on sharing information between trusted peers in a network to guard the network as a whole against intrusion attempts. We present initial ideas for running Indra over a peer-to-peer infrastructure to distribute up-to-date rumors, facts, and trust information in a scalable manner.

I. Introduction

A. Intrusion Detection Systems

Intrusion is the act or attempted act of using a computer system or computer resources without the requisite privileges, causing wilful or incidental damage. *Intrusion detection* involves identifying individuals or machines that perform or attempt intrusion. *Intrusion Detection Systems* (IDS) are computer programs that attempt to perform intrusion detection by comparing observable behavior against suspicious patterns, preferably in real-time. Intrusion is primarily a network based activity. With increasing global network connectivity, the topic of intrusion has gained prominence, spurring active research on efficient IDS.

Intrusion detection systems can be classified on the basis of a multitude of factors. Some factors significant to our project are listed below. [1] provides more and deeper information.

The work leading to this publication was performed while all authors were with Washington University in St. Louis.

Response to Intrusion: This may be passive or active. A passive system is content with just detecting intrusion, leaving its handling to a second, typically human, agency. On the other hand, an active system takes action, for example terminating network connections to a suspected host. Obviously, active systems can react more quickly and to more events, but open themselves up to denial-of-service attacks by over-reacting to deliberately triggered false alarms.

Source of audit data: The data to be examined could be network data like packet traces or host data like system call traces.

Data collection and processing: Data collection may be centralized or distributed. Again, this data may be processed centrally or at distributed locations.

In recent times, there has been a lot of interest in distributed schemes for intrusion detection. While the research community has been active in this area [2–8], most existing schemes are passive in the sense that they only implement the act of collecting information in a distributed manner. The controlling intelligence is centralized in the person of the system administrator(s) managing the administrative domain. Getting exactly the relevant information to this central entity is a difficult task that needs to achieve a fine balance between overloading the administrator or not providing enough information. Therefore, an autonomous system is needed to augment or eventually replace this central entity.

B. Outline of this paper

The motivations and current design of the Indra system are described in Section II. Section III discusses the deployment of Indra over peer-to-peer (P2P) systems. In Section IV, we discuss issues with trust and key distribution. In Sections V and VI, we propose a plugin mechanism that provides for dynamic extensibility in Indra. We discuss future and related work in Sections VII and VIII, respectively, and summarize in Section IX.

II. INDRA

Project Indra is named after an Indian God credited with a protective function. It also expands to *INtrusion Detection and Rapid Action*, which describes its goal and functionality with surprising accuracy, given that the acronym was retro-fitted.

A. Attacks on Immune Systems

Indra is an intrusion detection tool that takes a proactive and P2P approach to network security. It is often the case that attackers try out common exploits on different machines, hoping to stumble upon a machine on which a particular vulnerability is extant. Sometimes these attacks are detected and repulsed by intrusion detection software in place on a particular machine. But a persistent attacker, after many attempts [9], eventually manages to find a weak link in the chain. The broad goals of project Indra is to distribute such attempt information (gathered by the intended victim) among all interested peers in a P2P network. This allows the system to react, either proactively (e.g., by applying patches, temporarily disconnecting services, or both) or retroactively (e.g., disconnect machines that may have been compromised, to limit further damage).

The chance that at least one of the machines does notice an attack to which it is not itself vulnerable increases with the number of machines, the heterogeneity of the machines (operating systems and/or applications), and the level of currency of the applied security fixes. This makes it very attractive to have a system spreading such information quickly and widely.

B. Neighborhood Watch

Each interested host on the P2P network runs a special security daemon, the Indra daemon, which both watches out for intrusion attempts and also enforces access control based on its memory of earlier attempts. The P2P network needs to be reliable and trusted. This is achieved by applying trust management schemes such as the Web of Trust as known from PGP [10]. Extreme care must be taken when implementing the system not to open any security holes or opportunities for denial-of-service attacks.

Besides notifications occuring when immune systems see an attack on themselves (see above), it is also possible for other machines ("neighbors") sharing a network to detect other hosts as being under attack. This is particularly effective if the network is a shared medium, but the same effect can be achieved by installing Indra on network gateways or on a machine attached to a "snoop" port of a network switch. In particular, as shown in Figure 1, the following sequence of events could occur. Please note that

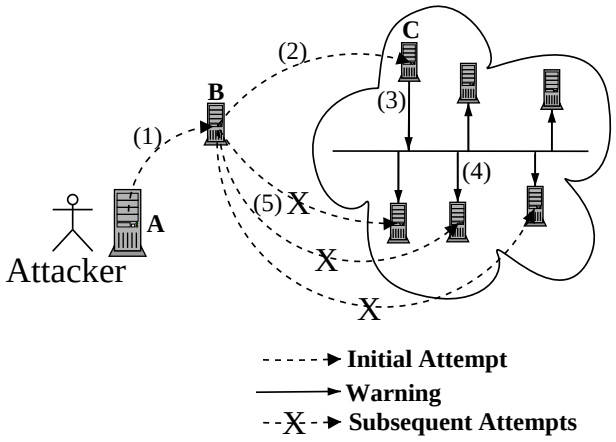

Fig. 1. Neighborhood Watch with Indra

in Figure 1, at least host C needs to be able to listen to B's network traffic.

1) The attacker on A finds the weak access point B in the network.
2) The attacker initiates attacks from B [1] to hosts in the trusted network to which the host C is connected. It is assumed that all hosts in the network, including C, run Indra daemons.
3) The Indra daemon at C detects the attack from B and then multicasts a secure warning message regarding B to its trusted neighbors.
4) Each Indra daemon receives the message from C, verifies its integrity and then places B on a 'blacklist' of suspected intrusion sources.
5) The attacker, having failed in his attempt on C, tries it out with other hosts in the same domain. These subsequent attacks are repelled straightaway by the forewarned hosts.

While this ideal situation is easy to spell out, it presents practical difficulties at various levels that have to be overcome first:

COMMUNICATION: How do the daemons communicate with each other? How do they transmit a message to all the other daemons? Some communication model has to be devised.

TRUST: How do the daemons trust messages and their senders? Obviously, messages have varying importance depending on who sends them.

POLICY: Suppose intrusion is suspected. How do the daemons react to it? Solutions can range from paranoia to indifference.

In the next few sections, we deal with each of these in turn.

[1] or a sequence of such Bs

III. Peer-to-Peer Communication and Indra

Indra relies on efficient group communication primitives in the underlying network in order to exchange intrusion information with peers. We argue that P2P systems, by providing fast and fault-tolerant primitives for search and data retrieval, provide an ideal platform on which Indra can be deployed.

As a case in point, we consider the *Scribe* [11] project, which overlays a topic-based publish-subscribe multicast mechanism on top of the *Pastry* peer-to-peer network [12]. In this scheme, Indra nodes are part of the Pastry network and communicate using Scribe groups, as shown in Figure 2.

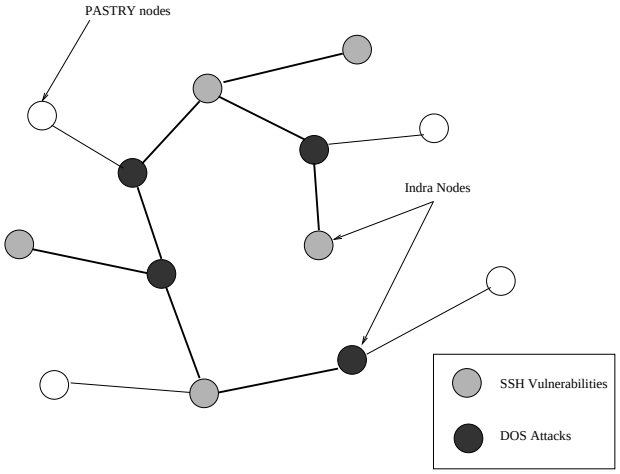

Fig. 2. Indra over *Pastry* and *Scribe*: The grey-colored nodes in the network are subscribed to messages related to SSH vulnerabilities, and the black ones to Denial-of-service attacks. Physically, both kinds of nodes are connected to the *Pastry* overlay network and communicate using the *Scribe* multicast protocol.

As an alternative to the deterministic multicast mechanisms outlined above, rumor-spreading models of communication have been proposed where each node propagates information to a randomized subset of its neighbors [13]. Such mechanisms are particularly relevant to Indra, since they enable Indra to be deployed on any peer-to-peer network without the additional overhead of creating multicast trees for each topic.

No matter what the actual communication substrate is, we believe Indra can effectively leverage the power and robustness of peer-to-peer networks. For example, the Gnutella [14] network allows end hosts to maintain multiple simultaneous connections with the network. This means that any group communication application built over Gnutella will naturally inherit the inherent fault-tolerance present in Gnutella and similar overlay networks.

A. Indra for Load-balancing

A significant advantage of distributed schemes in general, and Indra in particular, is that they may be used to balance the load on the detecting agent over many machines in the network. In modern high-speed networks, doing anything on a per-packet basis at link rates is getting increasingly difficult. Indeed, even a fairly simple operation like looking up the longest matching IP prefix becomes a bottleneck that calls for sophisticated techniques [15]. Therefore it is clearly infeasible to run a packet-scanning agent on a single bottleneck router. Schemes like Indra offer a way to distribute this load over hosts on the network. We are investigating efficient load-balancing schemes for this purpose, including randomized packet sampling techniques.

IV. Trust and Key Distribution

Trust is an important issue in an intrusion-detection system, more so in the absence of a centralized trusted authority to provide digital certificates (Certification Authority, CA). The usual decentralized alternate to central CAs is the web-of-trust model, where certifying happens among peers rather than from a central authority.

Our work on this is rather less concrete than that of Indra itself. In the prototype version, we rely on trusted keyservers from which Indra gets certificates for its peers. In a decentralized P2P system, variants of the *Web of trust* model from PGP [10] are more realistic. In this model, as shown in Figure 3, nodes are connected by trust relationships shown by edges, where edge weights represent degrees of trust. In reality, some nodes have pre-assigned trust values on entry, while trust values of other nodes must be computed based on their trust relationships. While there has been some work on trust metrics [16, 17] in a Web-of-trust model, this is currently an area of active research.

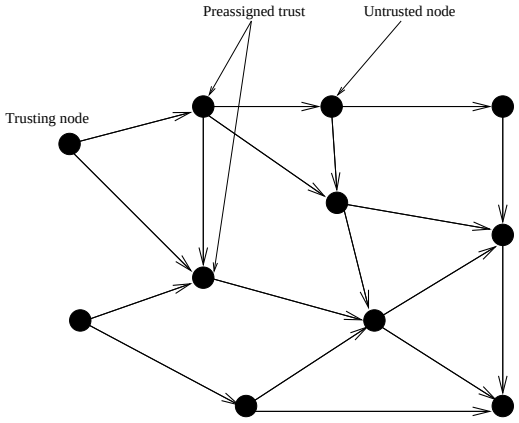

Fig. 3. Web of trust: the problem facing the node labeled "Trusting node" is to determine the extent to which it may trust untrusted nodes based on endorsements from other nodes which it does trust.

V. Indra Daemons

At the topmost level, all the functionality of Indra is achieved by a set of daemons which, in our implementa-

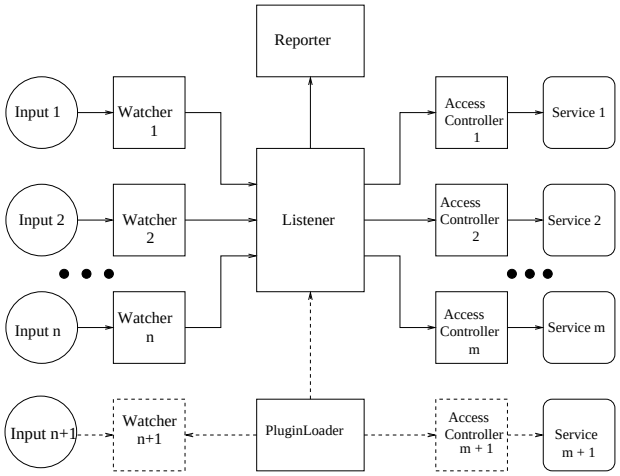

Fig. 4. Indra Daemons

tion, correspond to Java threads. These daemons belong to one of the following classes.

WATCHERS: These are the first level daemons that are on the outlook for any suspicious activity, either on the local system or over the network, for example multiple failed login attempts, port-scan attempts or suspicious system-call sequences.

ACCESS CONTROLLERS: These daemons provided controlled access to resources. The control is dynamic and depends on what the listeners tell them to do. When they get a warning against a particular user-id on a machine, they selectively filter out access to that particular (account, machine) combination. For determining accounts, it uses the IDENT protocol [18]. We are investigating enhancements to the IDENT protocol to incorporate digital signatures and the use of STOP [19].

LISTENERS: These are daemons that listen to the watchers. Listeners aggregate the warnings that are generated by the Watchers. Then based on the security level or any other policy dictated by the administrator, the listeners convey the warnings to the Access Controllers. Listeners are essentially selective filters that stand between the watchers and access controllers. If watchers were sense organs and access controllers limbs, the listener would be the central intelligence that drives motor function based on sensory input. For example, certain kinds of exploit attempts might result in vulnerable services being denied while other, presumably secure, services continue to operate normally.

REPORTERS: These daemons are responsible for communicating with other hosts, either receiving warnings and passing them on to the listeners or receiving aggregated warnings from listeners and passing them along the network to other hosts.

The daemons could be configured by the system administrator for different levels of security. For example, a host with critical information could be configured to deny all network connections to a machine which is identified as an originator of repeated failed logins. At another level, routers could run security agents that cut off packets that originate from a compromised machine, effectively isolating the machine from the network. Instead of taking it upon itself to make all these decisions, Indra provides a scaffold or framework that allows these options to be implemented by the administrator with ease.

VI. INDRA PLUGINS

Indra provides a mechanism by which additional daemons [2] can be plugged in at run-time into the Indra system. Whenever the administrator needs to change the security policy, either because a new exploit has surfaced or the security concerns have changed, she can write Java code that implements the necessary functionality and E-Mail or distribute it to interested peer daemons. These modules will be authenticated against the administrator's public key by the Plugin manager and then dynamically loaded into the daemon's address space.

We find that using Java for our implementation serves us well here. Code that compiles to native machine code, with its ability to forge pointers to arbitrary memory locations and to execute any combination of native machine instructions, is extremely difficult to audit or validate. Java, with its concept of a virtual machine as a sandbox, allows fine grained access control to resources, enabling different security policies for inbuilt code and code that is loaded over the network. This is analogous to executing Java applets securely inside the context of a browser.

VII. RESEARCH AGENDA

Indra is very much work-in-progress. We have a prototype implementation working, but it is too bare-bones to be useful in practice. For example, we use simple port-logging or failed-login counts as indicators of intrusion attempts. Overall, the fundamental contribution of Indra is not that of new intrusion detection techniques. Instead, we have tried to provide a framework that complements these techniques and help them maintain relevance in a massively networked scenario.

Ongoing research on Indra is on several fronts: The most important issue is that of trusting sources in a P2P system in the absence of centralized certifying authorities. We are investigating variations on the *Web of trust* model [10] which are appropriate for deploying Indra in a decentralized P2P manner. In addition, we will be using reliability measures as described in CONFIDANT [20].

[2] Watchers, Listeners or AccessControllers

Another area of interest is information propagation mechanisms for multi-party communication in P2P networks. We find that the publish-subscribe model described in [11] is closest to our work. Another area of relevant research is work on randomized rumor-spreading techniques [13] as a scalable alternative to deterministic flooding.

Currently, security advisories are written for system-administrators. However, it is a notorious fact that many system administrators are tardy in applying security patches. For example, more than a year after the discovery of the critical CRC32 bug [21], over 30% of the SSH servers still were vulnerable. Reasons that administrators do not update their systems include lack of time, but also fear of breaking existing applications and systems. We believe that a more selective approach like Indra may help keep systems secure even when updates have not been applied.

An interesting area of future research is on machine-readable advisories written in XML, which Indra daemons can autonomously act on.

Further, we are working on a standard and flexible interface to writing security plugins for Indra. Ultimately, this would enable the advisory agency to write plugin modules as soon as a vulnerability is detected, and place signed copies on the P2P network. As an alternative to P2P systems, an efficient multicast transport mechanism like SRM [22] or ALMI [23] could be used, if and when such mechanisms are widely deployed over the Internet. In any case, we predict turn-around time to be of the order of a few minutes, for machines distributed throughout the Internet.

VIII. Related Work

The idea of using distributed intrusion detection has been proposed with several variations over the past decade. Schemes have been proposed using distributed data collection and, in relatively fewer cases, distributed analysis agents.

An interesting approach to this problem using concepts of Immunology is [24]. The power of epidemics in networking has also recently gained interest [25]. The Distributed Firewall scheme [26] proposes a central access control access policy which is enforced by individual endpoints. The NADIR system [2] uses distributed data collection and centralized analysis by an expert system.

The GrIDS project [3] uses data source modules running in each host to extract information, which is used by graph engines to build a graph representation of network activity. GrIDS is again a purely a passive detection-based scheme, with corrective action presumably left to the system administrator. AAFID architecture [4] describes a distributed IDS based on multiple autonomous agents that can be added and removed from a system on the fly. There is no facility for automated handling of Intrusions, i.e., AAFID is a passive IDS.

The two schemes that are most closely related to Indra are Cooperating Security Managers (CSM) [5] and EMERALD [6]. CSM is an peer based IDS designed for use in a distributed network environment. Each CSM acts like a host-based local IDS for its host, while additionally cooperating with other CSMs without the use of a central controller. EMERALD is a powerful distributed IDS that is active and distributed. However, it does not seem to support on-the-fly plugin upgrades.

CITRA [27] proposes autonomic responses to distributed denial of service attacks by contacting upstream nodes in the path of the attack. An interesting area of future work is to implement CITRA-like responses to intrusion on top of the peer-to-peer mechanisms proposed by INDRA.

IX. Summary

As the global Internet becomes increasingly pervasive, computer intrusion and its prevention assumes greater importance. To be scalable with exploding network sizes, it is imperative that IDSs be distributed and self-maintaining.

In this paper, we argue the case of distributed intrusion-detection systems running over P2P networks. We describe the design of such a scheme, Indra, which promises to scale well under increasing network sizes and more determined attackers. We believe Indra, by leveraging the resilience of the underlying P2P network, has the potential to provide a robust intrusion detection system even in the face of concerted attacks.

At the frenetic pace at which software is written and deployed over the network, new vulnerabilities in networked systems crop up as fast as older ones are detected and plugged. In such a scenario, protection systems need to be pluggable to keep up with the latest bug-reports. Indra offers a scalable solution by providing for security plugins that can be loaded on the fly simultaneously by thousands of machines in an administrative domain.

References

[1] S. Axelsson. Research in intrusion-detection systems: A survey. Technical Report 98–17, Department of Computer Engineering, Chalmers University of Technology, December 1998.

[2] Judith Hochberg, Kathleen Jackson, Cathy Stallings, J. F. McClary, David DuBois, and Josephine Ford. Nadir: An automated system for detecting network intrusion and misuse. *Computers & Security*, 12(3):235–248, 1993.

[3] S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, J. Rowe, S. Staniford-Chen, R. Yip, and D. Zerkle. The design of grids: A graph-based intrusion detection system. Technical Report CSE-99-2, U.C. Davis Computer Science Department, January 1999.

[4] J. S. Balasubramaniyan, J. O. Garcia-Fernandez, D. Isacoff, E. Spafford, and D. Zamboni. An architecture for intrusion detection using autonomous agents. Technical Report 98/05, Purdue University, 1998.

[5] G. White, E. Fisch, and U. Pooch. Cooperating security managers: A peer-based intrusion detection system. *IEEE Network*, 10(1):20–23, 1994.

[6] P. A. Porras and P. G. Neumann. EMERALD: event monitoring enabling responses to anomalous live disturbances. In *Proceedings of the 20th National Information Systems Security Conference*, pages 353–365, October 1997.

[7] G. Helmer, J. Wong, V. Honavar, and L. Miller. Intelligent agents for intrusion detection. In *IEEE Information Technology Conference*, pages 121–124, September 1998.

[8] M. Crosbie and G. Spafford. Defending a computer system using autonomous agents. Technical Report 95-022, Dept. of Computer Sciences, Purdue University, Mar 1996.

[9] J. Howard. *An Analysis of Security Incidents on the Internet*. PhD thesis, Carnegie Mellon University, 1998.

[10] William Stallings. Pretty Good Privacy. *ConneXions*, 8(12):2–11, December 1994.

[11] Antony I. T. Rowstron, Anne-Marie Kermarrec, Miguel Castro, and Peter Druschel. SCRIBE: The design of a large-scale event notification infrastructure. In *Networked Group Communication*, pages 30–43, 2001.

[12] Anthony Rowstron and Peter Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, Heidelberg, Germany, November 2001.

[13] Richard M. Karp, Christian Schindelhauer, Scott Shenker, and Berthold Vöcking. Randomized rumor spreading. In *IEEE Symposium on Foundations of Computer Science*, pages 565–574, 2000.

[14] The Gnutella Network. http://www.gnutella.com.

[15] Marcel Waldvogel, George Varghese, Jon Turner, and Bernhard Plattner. Scalable high-speed prefix matching. *Transaction on Computer Systems*, 19(4):440–482, November 2001.

[16] Ueli Maurer. Modelling a public-key infrastructure. In *ESORICS: European Symposium on Research in Computer Security*. LNCS, Springer-Verlag, 1996.

[17] Michael K. Reiter and Stuart G. Stubblebine. Path independence for authentication in large-scale systems. In *ACM Conference on Computer and Communications Security*, pages 57–66, 1997.

[18] Identification protocol. Internet RFC 1413, http://www.faqs.org/rfcs/rfc1413.html, 1993.

[19] Brian Carrier and Clay Shields. A recursive session token protocol for use in computer forensics and tcp traceback. In *Proceedings of Infocom 2002*, pages 1540–1546, June 2002.

[20] Sonja Buchegger and Jean-Yves Le Boudec. Performance Analysis of the CONFIDANT Protocol: Cooperation Of Nodes—Fairness In Dynamic Ad-hoc NeTworks. In *Proceedings of IEEE/ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC)*, Lausanne, June 2002. IEEE.

[21] Roman Danyliw, Chad Dougherty, and John Shaffer. Exploitation of vulnerability in SSH1 CRC-32 compensation attack detector. Incident Note IN-2001-12, CERT, November 2001.

[22] Sneha Kumar Kasera, James F. Kurose, and Donald F. Towsley. Scalable reliable multicast using multiple multicast groups. In *Measurement and Modeling of Computer Systems*, pages 64–74, 1997.

[23] Dimitris Pendarakis, Sherlia Shi, Dinesh Verma, and Marcel Waldvogel. ALMI: An application level multicast infrastructure. In *Proceedings of the 3rd USNIX Symposium on Internet Technologies and Systems (USITS '01)*, pages 49–60, San Francisco, CA, USA, March 2001.

[24] S. Forrest, S. Hofmeyr, and A. Somayaji. Computer immunology. In *Communications of the ACM, (submitted Dec. 1996)*, 1996.

[25] Werner Vogels, Robbert van Renesse, and Ken Birman. The power of epidemics: Robust communication for large-scale distributed systems. *Computer Communication Review*, 33(1), January 2003. Proceedings of HotNets-I.

[26] S. Ioannidis, A. Keromytis, S. Bellovin, and J. Smith. Implementing a distributed firewall. In *Proceedings of Computer and Communications Security (CCS)*, November 2000.

[27] Dan Sterne et al. Autonomic response to distributed denial of service attacks. In *Proceedings of RAID 2001*, October 2001.