Active Systems Management: The Evolution of Firewalls

William A. Arbaugh*

Department of Computer Science
University of Maryland
College Park, Maryland 20742
waa@cs.umd.edu

Abstract. For over twenty-five years, the security community has focused on improving security, and yet information systems remain as vulnerable as ever (perhaps more so). A perfect example of this is the (first intrusion occurred on July 10, 2001 and peaked on July 19, 2001) code red worm which automatically exploits a vulnerability in Microsoft's Internet Information Server (IIS). Following a successful exploitation of a host, the worm propagates by scanning for other vulnerable hosts to infect. Nine days later, approximately 250,000 hosts were reported as infected. The unfortunate truth about the code red incident is that ALL of the intrusions should have been prevented. The incidents should have been prevented because Microsoft released a patch/fix for the vulnerability on June 18, 2001– almost a month before the beginning of the code red worm's first release. The fact that information systems remain as vulnerable (or more so) than they were over 25 years ago when research began implies that a paradigm shift is in order. In this paper, I will motivate the need for a paradigm shift by describing current problems, and the trends which will only amplify those problems. I will then introduce the notion of Active Systems Management where both management and protection are provided on a per host basis.

1 Introduction

A fundamental fact in computer and network security is that there never can be a one hundred percent assurance that a computer system is *trusted*. By *trusted*, I mean that an object is *trusted* when it always operates as expected by design and policy [1]. Ken Thompson described very clearly one of the many issues involved in determining if a system is *trusted* in his Turing Award speech in 1984 [2]. For over twenty-five years, the security community has focused on technology, and yet information systems remain as vulnerable as ever (perhaps more so). A perfect example of this is the (first intrusion occurred on July 10, 2001 and peaked on July 19, 2001) *code red worm* which automatically exploits a vulnerability in Microsoft's Internet Information Server (IIS). Following a successful exploitation of a host, the worm propagates by scanning for other vulnerable hosts to infect.

* This work will be supported by a NSF Career Award beginning September 1, 2002.

Nine days later, approximately 250,000 hosts were reported as infected with the worm [3]. The unfortunate truth about the *code red* incident is that ALL of the intrusions should have been prevented. The incidents should have been prevented because Microsoft released a patch/fix for the vulnerability on June 18, 2001– almost a month before the beginning of the *code red* worm's first release.

But, this poor management is only one of the many problems facing organizations today. Another is the over reliance on firewall technology to protect the organization. These two problems (and others) have created a situation where information technology is more vulnerable than ever [4].

1.1 The Problems with Patching

There is an old saying within the security community that– *"90% of all intrusions are because of KNOWN vulnerabilities."* While anecdotally this saying was believed accurate, there were no empirical studies supporting or contradicting the statement. Recently, a study demonstrated that the statement is most certainly true. In fact, a stronger statement can be made that "approximately 99% of all reported intrusions occur because of KNOWN and FIXED vulnerabilities" [5, 6]. Furthermore prior to this research, the general belief was that the number of intrusions decayed after a fix or patch for a vulnerability was released, see Figure 1 [7, 8]. Unfortunately, that belief is incorrect, and the real histogram is shown in Figure 2 which has a significant positive skew.

Essentially, the problem is that the attackers react faster to new vulnerabilities than the defenders (IT managers). As a result, known vulnerabilities are readily available for easy exploitation by the attackers. This situation is similar to that of an irregular military confrontation, and a well established theory exists that describes the phenomenon entitled the *Observation, Orientation, Decision, and Action* or OODA loop [9, 10]. This theory put forth by Col. John R. Boyd essentially states that whomever can react to changes the fastest in a confrontation, i.e. has a tighter OODA loop, will prevail. This is certainly the current case in computer security recent research has shown [5, 6].

While studies, anecdotal evidence, and press reports have demonstrated the increasing vulnerability of information technology, information security research is currently primarily focused on the underlying *security technology* rather than the secure management of the information technology. Yet, the tremendous growth in the use of information technology and user mobility and its rate of change creates a configuration and system management nightmare that amplifies existing security problems. Unfortunately, current approaches for solving this complex problem are *ad hoc*, do not scale, and have not focused on the importance of security.

1.2 An Over Reliance on Firewalls

An element of current best practice is the isolation of an enterprise's information resources and the canalization of ALL external connections through one (or a few) well defined paths. Along each of these paths, a firewall is established to

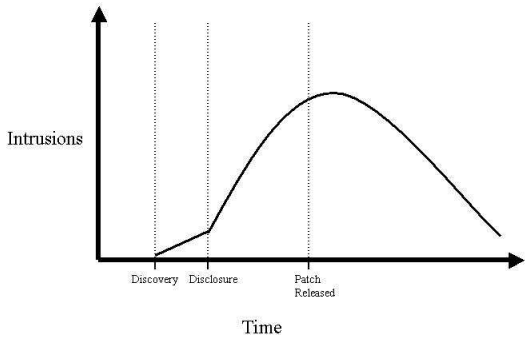

Fig. 1. Intuitive histogram of intrusion rates

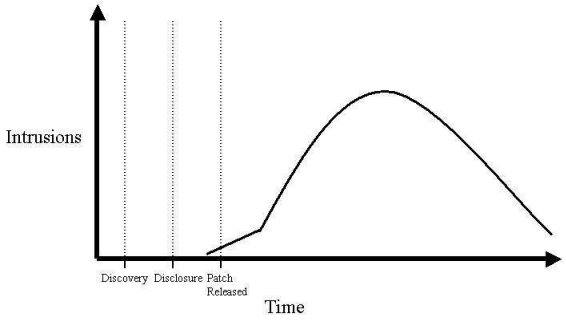

Fig. 2. Actual histogram of intrusion rates

mediate all connections to ensure that only authorized connections enter and leave the enterprise. A firewall as defined by Merriam-Webster's is "a computer or computer software that prevents unauthorized access to private data (as on a company's local area network or intranet) by outside computer users (as of the Internet)." Firewall's began their climb to ubiquity in the early 1990's as a counter-measure to the tremendous increase in malicious activity on the Internet, and they worked well. However, the increase in "active content", e.g. ActiveX and JavaScript, utilized by client software, increases in user mobility, and peer-to-peer technology are now making the traditional firewall less effective. As a result, the security community must find ways to revolutionize best practice rather than continuing to evolve it.

While firewalls initially provided a significant increase in protection for organizations, the evolution of technology (including firewall technology) has evolved such that firewalls no longer provide the same degree of protection. For instance, the tremendous increase in "active content" that a client receives via electronic mail and through surfing the Web make preventing the introduction of malice into the enterprise difficult– if not impossible. While the developers of the "active content" make claims as to the safety of the content's execution environment, the complexity of designing and implementing such environments ensures that security vulnerabilities will exist. These vulnerabilities will be found, and regardless of the disclosure method used by the bug's finder- the bug will eventually be patched by the vendor. However, now ALL of the machines behind the firewall MUST be patched to prevent the future exploitation of the vulnerability- something we now know seldom happens [5]. As a result, the enterprise remains vulnerable and the utility of the traditional firewall diminishes. Furthermore, the mobility of end users continues to grow dramatically, and these mobile users need access to the enterprise's data and infrastructure while out of the office. To address this problem, virtual private networks (VPN) were created.

A VPN creates a secure tunnel between two communicating parties- securely extending the infrastructure of the enterprise to the mobile user either at home or on the road. In essence, the VPN permits the mobile user to by-pass the firewall in a controlled fashion. While VPN's are highly effective at protecting the transmission of data between the two communicating end points, VPN's can not protect the mobile user's laptop or home computer, and these computers become a potential vector for malice into the enterprise because these computers are essentially "dual homed" to the enterprise's intranet and the Internet. Since most user's are unable to reliably protect their system, the home computer or laptop presents the path of least resistance into the enterprise's intranet for an attacker. A fact that has not gone un-noticed by attackers as several well publicized breaches of large corporations have occurred this way.

A truism in security is that an attacker, much like current, will always take the path of least resistance, and while the previous two technical problems provide a means for by-passing the protection offered by a firewall, the biggest problem by far is that of the users. Users, like attackers, will also take the path of least resistance. Firewalls and other security mechanisms usually impose a duty upon the user to "do something different" or something more than the usual, and some users believe this security duty impairs their ability to perform their job. Therefore, they circumvent the obstacle-creating a "short" in the security protections. One of the easiest ways for a user to by-pass a firewall is to create an unauthorized external entry point by adding a modem or wireless LAN access point (AP) to their host on the enterprise's intranet. Now, the user can call up their office phone or connect to the AP when ever they want under their terms. Unfortunately, attackers can do so as well. Furthermore, managers can be just as culpable as the users by relying on the protection of the firewall too much. In this case, the managers neglect the configuration management of the internal machines-creating a "hard exterior and a chewy interior" for attackers.

Finally, the explosive growth in peer to peer technology has created yet another method for by-passing firewalls to the point where some of the file sharing technology actually creates a private virtual network for the company that produced the file sharing technology. In essence, best practice today digs a moat around the enterprise and the firewall acts as the draw-bridge. This worked well for some time, and still offers some protection. But as in the history of warfare, changes in technology have made the moat less effective.

While distributed peer to peer technology has a created a potential security problem, it may also be pointing the way to a solution. For years now, security researchers have focused on two distinct problem areas: infrastructure or enterprise security, and ad-hoc or security for ubiquitous computing. I believe the time has come where the two areas must merge, and every information device must be able to protect itself via a series of defensive mechanisms, e.g. defense in depth. This idea is not entirely new. In 1999, Steve Bellovin and others recognized that in some cases the topology of the network would not permit a single entry protection profile, and he proposed the novel idea of distributed firewalls where each information system enforces a centralized policy [11]. This approach, however, begs the management problem with the potential for making an already difficult problem more difficult. What then is the answer? Perhaps peer to peer systems management is the answer? But, what is peer to peer system management, and why is it important? In this paper, I will describe the notion of Active Systems Management and it's challenges.

2 Active Systems Management

There are several aspects that make Active Systems Management (ASM) a novel approach to security. First, it is one of the first research projects to examine security from a management perspective rather than a solely technical one. The second is the view of management from a distributed or peer to peer sense rather than centralized (although ASM could be managed centrally as well), and finally ASM takes the position that not only should information systems and devices be able to protect themselves– they must be able to reliably determine when they have been compromised.

In the next few sections, I present several of the challenges that ASM must solve to become viable.

2.1 Formalization of Active System Management

One method of formalizing secure distributed systems management by modeling the life cycle of a managed system with a finite state machine. This novel approach captures the dynamic nature of such systems in a formal fashion without any loss of generality.

Host State Model An initial finite state machine for a host involves only three states: *Hardened, Vulnerable, and Compromised.*

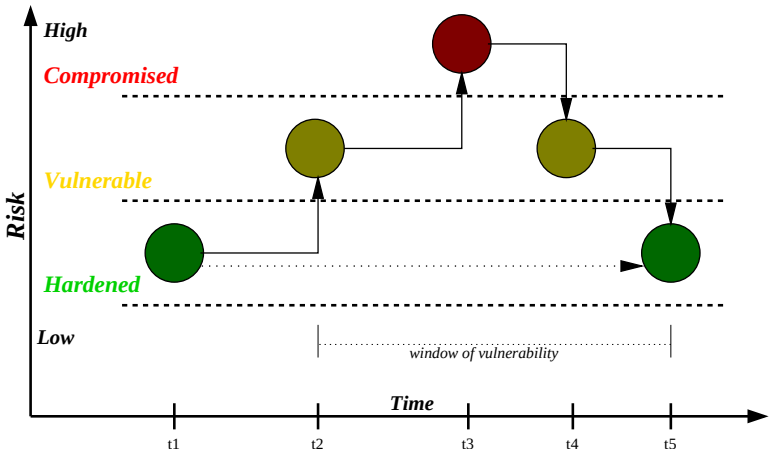

Fig. 3. An example of host transitions over time

The first state is a *hardened* state. In this state, the system is configured (including access control lists and other security mechanisms) and patched to prevent all **known** vulnerabilities. *Secure* is not used to define this state because we cannot prove the system prevents all attacks, i.e. we may be vulnerable to attacks unknown to the maintainers. The system should begin life in this state, i.e. shipped by the vendor. Unfortunately, this is often not the case.

At some point during the system's life, a vulnerability will be discovered (or a configuration error will occur) which may permit a security violation to occur. At this point, the system transitions into a *vulnerable* state. The system is vulnerable– but not exploited. Once in the vulnerable state, the system can return to a hardened state through a configuration change or the application of a patch correcting the vulnerability. Or in the worst case, the system is exploited by the vulnerability and enters the *compromised* state. Once a system enters the compromised state, transitioning back to a hardened state becomes extremely difficult because the actions of the attacker are usually unknown. The actions are usually unknown because the attacker obtains system level privileges and erases logs etc. to cover their tracks. Figure 3 depicts a series of these transitions, and the state machine for the life of a host is shown in Figure 4.

The sum of the time that the host is either in the vulnerable or compromised state is considered the *window of vulnerability*, and is depicted in figure 3, and the overall risk of a system in this model is described by equation 1,

$$r_{total} = \sum_{k=0}^{n} R(S, t_k) \tag{1}$$

where r_{total} is the total risk of a system from time t_0 to time t_n, and the function $R : S\ x\ t \rightarrow r$ returns the risk associated for a system, S, at some time t. A key

observation here is that the total risk to a system can never be zero because the hardened state includes some risk.

The goal of ASM is to minimize the value of equation 1 to as small as possible. The obvious approach to meet this goal is to reduce the overall time that a system remains in the *vulnerable* or *compromised* state– reducing the *window of vulnerability* of the system to as small as possible resulting in a significant increase in the work factor of an attacker.

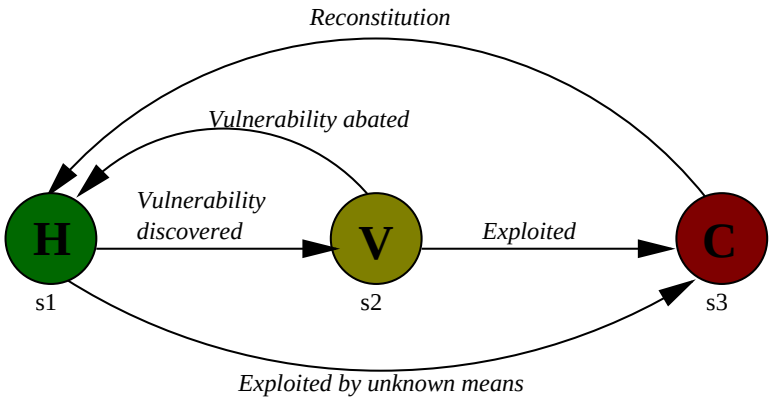

Fig. 4. Host state machine

Ideally, the transition from *Hardened* to *Compromised* will never occur, but prudence dictates that we plan none the less.

Once the finite state machine is well defined, we must answer two important questions:

1. *What is the current state of the system?* The verification that a system is in a secure state while the system is in operation is one of the most important goals of system administrators. It is also one of the more difficult. Strong cryptographic configuration management provides an excellent mechanism for answering this question. Applications such as Tripwire have had a strong track record of identifying intrusions by detecting un-approved integrity violations [12]. Unfortunately, applications such as Tripwire assume that the operating system is trusted. This can result in false negatives when a system is compromised and the operating system modified to provide incorrect responses. In fact, there exists several attack tools which defeat integrity tools such as Tripwire. The use of an out-of-band / independent auditor or Oracle solves this problem, see section 2.2.
2. *How to transition to the hardened state?* If the state of the system is determined to be in either the *Vulnerable* or the *Compromised* state, then the minimal set of operations must be found which transition the system back

to the hardened state. A first order estimate can be found by examining the changes to the system's configuration, found by the out-of-band auditor. Further research, however, is required to ensure that this is sufficient, but early indications are that is.

The next section describes several possible methods for answering the above questions.

2.2 Methods for Active System Management

Current approaches for secure distributed systems management are either proprietary, *ad hoc*, or fail to scale. As a result, wide-scale distributed systems are difficult (if not impossible) to manage in a secure manner. Coupling this with the lack of trained system administration personnel creates a significant problem in maintaining information systems in a secure posture.

The process for ASM will involve several distinct phases. The first is obviously communications. In ASM, the communications can either be peer to peer, centralized, or a combination of both. It all depends on the policy of the managed device. Once the device begins receiving management instructions, the instructions both the source and data integrity of the instructions must be validated. The instructions themselves must also be validated in a similar fashion as mobile code is currently validated.

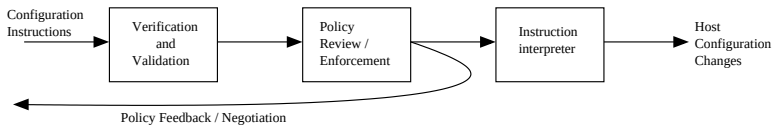

Fig. 5. Instruction flow at a managed host

Policy Negotiation In a large and diverse population, there will be sites that do not wish, for a number of reasons, to permit fully automated management of their information systems. Therefore, we must ensure that the system supports and achieves the same results when used in a fully automatic mode, a fully manual mode, or in a mode between the two. Anything less, and the system adoption rate will suffer. Additionally, drastically different security policies exist throughout large organizations. In most cases, elements of organizations will communicate only amongst those that share a common security policy. In other cases such as in a collaborative work, however, elements and organizations with differing security policies must communicate to complete a mutually beneficial objective. A successful system will permit the members of a managed set (a web or confederation if you will) to have different security policies significantly easing

a current management burden of maintaining multiple management domains. Accomplishing these research goals, will require leveraging off of the research efforts within the *Trust Management* area [13, 14, 11]. In fact, the process shown in Figure 5 is a variation of the process used in trust management.

Enforcement of Configuration Management One aspect not addressed by current approaches to systems management is an auditing capability. Systems such as cfengine [15] send a description to each target system specifying the desired state the system. It is up to the target system to ensure that it meets the specification. If the security of the system has been breached, then it cannot be trusted to make the required state changes nor provide an accurate report on its current state. Out-of-band security audits provide a means to address this situation.

Applications such as Tripwire can assist in such audits, but these tools are usually difficult to operate, and may provide incorrect results, i.e. a false negative as was the case with the authors of the Internet Auditing Project [16]. The problem with integrity applications is that they depend on the integrity and proper operation of the operating system, i.e. these applications assume that the operating system always operates correctly. When this assumption is not valid the integrity applications cannot provide a reliable result, and as a result the applications provide a false negative. Tools such as those that can be found at RootShell.com provide novice attackers with the capability to defeat integrity tools that rely solely on the operating system. Additionally, most integrity applications require a database of pre-computed values. The maintenance and protection of this database presents additional challenges to effective and secure management on a wide scale.

The approach in this proposal provides a solution to the traditional problems, described above, with integrity and configuration management using Out-of-Band Cryptographic Configuration Management. The solution leverages digital signature technology to provide integrity protection for file systems with an out-of-band verification process that does not depend on the underlying operating system. The resultant system provides extremely strong integrity guarantees and configuration management– detecting modifications to approved objects as well as detecting the existence of un-approved and thus unsigned objects. This is accomplished without ANY modifications to the host operating system. As a result, the system is equivalent to an independent auditor capable of detecting problems in near real-time.

We are currently implementing the out-of-band *near real-time* auditing capability using a stand-alone embedded processor, or Oracle. The Oracle can either passively monitor the system by examining the system input-output bus for integrity violations of the host's file system, or the Oracle can actively monitor the system by issuing IO requests in the same manner as the audited system. The former approach can be taken when performance is an issue, and the latter approach when security is considered essential. Of course, approaches combining the two are possible as well. Additionally, it may be possible to host the security

management software entirely on the embedded Oracle. This will provide further assurance that the system is operating as expected [17].

When non-compliance problems are found by an auditor, it reports the problems to the management authorities via the network who can work to resolve the issue based on policy. The non-compliant system could be isolated, repaired, or monitored more closely to determine the extent of the problem.

The following paragraphs describe the issues involved with each transition within the current state machine model.

Hardened to Vulnerable When a flaw is discovered with security or surviability implications for the managed system, a transition occurs from the *hardened* state to the *vulnerable* state. The only means for determining when this transition occurs is with an effective "intelligence" apparatus, i.e. the monitoring of security related mailing lists such as *BugTraq* and hacker IRC channels. For Active System Management to be effective, this information must be timely. Delays will only increase the length of time that ALL of the managed systems remain in the vulnerable state. These "intelligence" gathering aspects are not part of ASM, however, they are an important aspect of the work.

Once a vulnerability is discovered, a series of steps must be accomplished as quickly as possible:

1. If intrusion detection systems are actively managed, then a *signature* for the exploitation must be developed and pushed to the sensors to ensure detection of any immediate exploitations of the vulnerability.While not specifically a focus of ASM, the active management of intrusions detection systems (see above) can greatly increase the assurance on an organization.
2. If a vendor patch is available (and they usually are [5]), then it must be tested to ensure correctness. If a vendor patch is not available, then a temporary configuration change or organizational patch must be considered.
3. Once a patch or configuration change is tested and approved, it becomes available to the managed systems for installation. If the priority of the patch is high (based on the severity of the flaw and the rate of intrusions [6]), a *priority* message can be sent to the systems to inform them that a high priority configuration change is available.

Hardened/Vulnerable to Compromised The most effective way to determine when a host transitions from a hardened to a vulnerable or compromised state is to identify the first order effects of the attacks such as integrity compromises or configuration changes. The out-of-band-auditor performs these checks in an extremely high assurance manner.

Vulnerable to Hardened Transitioning a host from a *vulnerable* state back to a *hardened* as quickly as possible without compromising reliability is not as straight forward as one might think– especially in a heterogeneous environment.

A number of issues must be considered and addressed such as the resources available on the managed system, i.e. some systems may not have enough memory or storage space to repair the system. This is one of the major reasons for implementing the system as shown in Figure 5. This permits each managed element to provide feedback so that the configuration instructions can be adapted to fit each managed element.

Compromised to Hardened The usual course of action with a compromised system is to completely rebuild the system's software from scratch– possibly formatting the hard drive. This grievous approach is time consuming and may cause the loss of important data. However, is this approach sufficient? Could the firmware have been compromised as well [18]?

3 Conclusions

The current state of information security is clearly not good. Unfortunately, the reasons for this terrible state of affairs are many. What is also clear is that one can never have one hundred percent effective security, i.e. there is no *silver bullet*. Additionally while firewalls were once extremely effective at preventing intrusions, changing technology and user usage patterns are making firewalls less effective. The goals of Active Systems Management are to find ways to manage and protect heterogeneous systems better in a fashion that easily applies to current and future information systems and devices.

References

1. P. G. Neumann, "Architectures and Formal Representations for Secure Systems," Final Report. SRI Project 6401 A002, SRI International, October 1995.
2. K. Thompson, "Reflections on Trusting Trust," *Communications of the ACM*, vol. 27, pp. 761–763, August 1984.
3. CERT, "*Code Red* Worm Exploiting Buffer Overflow In IIS Indexing Service DLL." `http://www.cert.org/advisories/CA-2001-19.html`, July 2001.
4. May 1998. Consensous of attendees at IEEE Security and Privacy Conference during panel session.
5. W. A. Arbaugh, W. L. Fithen, and J. McHugh, "Windows of vulnerability: A case study analysis," *IEEE Computer*, vol. 33, pp. 52–59, December 2000.
6. H. Browne, W. A. Arbaugh, J. McHugh, and W. L. Fithen, "A trend analysis of exploitations," in *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, pp. 214 – 229, May 2001.
7. B. Schneier, "Closing the Window of Exposure: Reflections on the Future of Security." SecurityFocus.com, 200. http://www.securityfocus.com/templates/forum_message.html?forum=2&head=3384&id=3384.
8. J. Howard, *An Analysis Of Security Incidents On The Internet: 1989 – 1995*. PhD thesis, Carnegie Mellon University, April 1997.
9. J. R. Boyd, "The Essence of Winning and Losing." http://www.belisarius.com/modern_business_strategy/boyd/essence/eowl_frameset.htm, June 1995. Revised January 1996.

10. J. R. Boyd, "Organic Design for Command and Control." http://www.d-n-i.net/FCS_Folder/boyds_discourse/c&c.pdf, May 1987.
11. S. Ioannidis, A. D. Keromytis, S. Bellowivn, and J. M. Smith, "Implementing a Distributed Firewall," in *Proceedings of the ACM Computer and Communications Security Conference*, pp. 190 – 199, 2000.
12. G. Kiim and E. Spafford, "Experience with tripwire: Using integrity checkers for intrusion detection," in *System Administration, Networking, and Security Conference III*, USENIX, 1994.
13. M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized Trust Management," in *Proceedings of the IEEE Security and Privacy Conference*, 1996.
14. M. Blaze, J. Feigenbaum, and M. Strauss, "Compliance Checking in the PolicyMaker Trust Management System," in *Proceedings of the Financial Cryptography Conference*, 1998.
15. M. Burgess and R. Ralston, "Distributed Resource Administration Using cfengine," *Software-Practice and Experience*, vol. 27, no. 1083, 1997.
16. L. Siri, "The Internet Auditing Project (beta 1)." http://www.viacorp.com/auditing.html.
17. J. Molina and W. Arbaugh, "Using Independent Auditors as Intrustion Detection Systems," Tech. Rep. CS-TR-4387, University of Maryland Department of Computer Science, July 2002.
18. CERT Coordination Center, "Frequently Asked Questions About the CIH Virus." http://www.cert.org/tech_tips/CIH_FAQ.html, May 1999.