Diversity as a Defense Strategy in Information Systems

Does Evidence from Previous Events Support Such an Approach?

Charles Bain, Donald Faatz , Amgad Fayad, Douglas Williams
The MITRE Corporation 1820 Dolley Madison Blvd McLean Va 22102 USA

Key words: Security, survivability, diversity, intrusion and fault tolerance

Abstract:

One of the challenges facing computer systems is resisting attack and compromise in a networked environment. Today's computing environment is fairly homogeneous, due to a relatively small number of operating systems and application functions running on the vast majority of computers. This environment allows attackers to focus their efforts on the few types of systems deployed. Once an exploit is found, the exploit is effective against a very large number of systems running the same software. The large number of attack methods available on hacker Web sites demonstrates the ease with which attackers can exploit this homogeneous environment. This paper examines several widespread computer attacks to understand the effect of diversity on maintaining the integrity, and hence survivability, of information systems.

1. INTRODUCTION

One of the challenges facing computer systems is resisting attack and compromise in a networked environment. Today's computing environment is fairly homogeneous, due to a relatively small number of operating systems (e.g., variants of UNIX or Microsoft Windows) and application functions (e.g., networking based on TCP/IP) running on the vast majority of computers. This environment allows attackers to focus their efforts on the few types of systems deployed. Once an exploit is found, the exploit is effective against a very large number of systems running the same software. The large number of attack methods available on hacker Web sites demonstrates the ease with which attackers can exploit this homogeneous environment.

Most systems run similar software and/or support common services. If systems were different, they might have an additional defense against attacker exploits: a vulnerability discovered in one system might not be effective in other systems if the systems are different in ways that avoid the vulnerability. Additionally, this diversity would increase the effort required to compromise systems, since each system would be a unique environment for an attacker to work against. This increase in effort could reduce the number of exploits discovered (because of the additional effort required) and perhaps decrease the attractiveness of exploiting systems.

Diversity as a defense is illustrated in attacks on systems. A typical attack exploits a system and a specific vulnerability: different systems are not (directly) affected. An attack effective against Microsoft Outlook does not affect UNIX mail applications. However, different computer systems support common services, such as the TCP/IP networking protocol used for Internet access, or common applications, such as the Netscape or Internet Explorer web browsers. Thus, the advantages gained through diversity are offset by systems with a common point of failure.

This paper examines several widespread computer attacks to understand the effect of diversity on the survivability of attacked systems. The described attacks are:

- the Morris worm, which spread by exploiting vulnerabilities in TCP/IP capabilities
- the Melissa virus, which infected using the macro capabilities of a Microsoft Word attachment to e-mail
- the LoveLetter worm, which infected using a Visual Basic script attached to e-mail
- the Denial of Service attacks against high-profile web sites from a network of compromised “slave” systems

The attack methods and effectiveness are described. In the conclusion of the paper, the role of diversity in the survival of systems is discussed.

2. THE MORRIS WORM

Introduction

On Wednesday, November 2, 1988, at 5:01:59 P.M. E.S.T. a worm was released on the Internet [20]. It was brought under control and eliminated from most machines 48-72 hours later [19, 26]. This self-propagating worm easily spread by exploitation of well-known vulnerabilities that had not been closed in the victim systems.

Technical Summary

The Morris Worm used four main methods for spreading:

- fingerd gets() buffer overflow: Only 4.3BSD VAX machines suffered from this attack [19]. SunOS did not suffer, causing a core dump, only because of different required offset on the stack [18, 21, 26]. Ultrix, for example, was not vulnerable [8].
- Sendmail DEBUG option: Mostly Berkeley derived Unixes, but also other varieties of Unix [25, 26]. SunOS binary releases had this mode [8]. DEBUG was enabled as the default for 4.2BSD, 4.3BSD and derived SunOS, while the

commercial release of Ultrix did not have DEBUG enabled as a default [8].

- Trusted logins using .rhosts and /etc/hosts.equiv with rexec and rsh: This affected networking code based on BSD extensions [25]. These are inherently insecure functions.

- Passwords where /etc/passwd file was not shadowed.

Once security was breached, a bootstrap program was sent to the compromised machine, which was compiled after its transfer. This program was then executed and proceeded to copy over two object files (plus the bootstrap source code), one for the VAX BSD and one for the Sun-3 SunOS. The bootstrap program linked the appropriate file against the C library to produce an executable worm. Hence, the worm "supported" only a BSD UNIX and derived operating systems in use at the time of release. There were unused provisions in the worm code to transfer an additional 17 files [17], indicating additional targets may have been planned.

It was estimated that approximately 75 percent of the computers then attached to the Internet used some version of Unix [27], but the worm only affected code that included 4.2 or 4.3 BSD derivatives like SunOS [21]. Furthermore, the worm only propagated over TCP/IP and not UUCP, X.25, DECNET, or BITNET [21]. The worm did not infect System V systems unless they had been modified to use Berkeley network programs like sendmail, fingerd and rexec [21].

Extent of Infection

In November 1988 it was estimated that there were approximately 60,000 computers worldwide on the Internet [13, 25], composed of over 500 unclassified national, regional and local networks [27]. The NSF estimated that there were over half a million Internet users [27] at the time.

There are no official estimates of the number of computers attacked, in part because no one organization is responsible for obtaining such information. The actual number of systems infected is

impossible to determine, but it's worthwhile to examine the frequently quoted figures.

The first estimate came on Thursday, November 3, 1988, when in the late evening MIT held a press conference stating that they had suffered an estimated 10% infection rate of the 2,000 hosts belonging to MIT. The infection rate was a guess at the time and was given when the Internet was still under attack. The press extrapolated this percentage to the entire Internet and concluded that 6,000 machines, [20, 27] of the 60,000 estimated to comprise the Internet at that time, were infected.

However, not all sites have the same proportion of vulnerable machines as MIT. A Harvard University researcher who queried users over the Internet contends that a more accurate estimate would be between 1,000 and 3,000, or 2% to 5% of the computers infected [27]. Other estimates at the time ranged from 2,000 to 6,000 (3% to 10%), but when the situation stabilized, consensus among published papers centered around 2,000 to 4,000 (3% to 7%) [4, 5, 16, 25].

Remarks

One of the problems with the available information is that the extent of infection of vulnerable machines is unknown. If this information were available, it would be possible to map this proportion into the total number of Internet hosts to yield an estimate of infection that would have occurred if the Internet had been homogeneous.

Security is a tradeoff, a measurement of the resolve of the attacker and defender to commit resources to gain an advantage. In a homogeneous computing environment, less expenditure of resources will be required to defend the system. Similarly, a lesser commitment of resource is required to attack the system as Shoch and Hupp found when developing their worm [22]. In a heterogeneous system, the reverse is true for both the defender and attacker. Hence, the issue is whether the defender or the attacker has the resolve to commit greater resources to the problem.

In the worm example, at the time of release, the attacker had only committed resources to permit the attack of a subset of BSD based systems. However, 17 additional operating systems may have been considered as targets [23]. It is arguable that if the worm author had committed the resources to the attack, the Internet would truly have been brought down.

3. THE MELISSA VIRUS

The Melissa virus (W97M.Melissa.A) was released into the Internet in March 1999. Melissa is a Microsoft Word Macro virus which uses electronic mail (e-mail) to spread itself to additional systems; however, it carries no malicious payload.

Technical Summary

The Melissa Virus spreads by attaching an infected Word document to an e-mail message. Recipients who open the attached document in Word experience two side effects:

- Word documents created after the infection are also infected with the virus.
- E-mail addresses from their address book are used to further spread the virus.

Melissa also reduced the level of security warnings displayed to Word users and modified the Windows Registry to indicate its presence so future re-infections would have no additional affects.

Extent of Infection

It is difficult to assess the overall extent of the infection. None of the sources located to date could say with any certainty how many systems were infected. Computerworld [17] reported that 80 percent

of the 150 organizations that contacted Network Associates for assistance were infected. This article also reports that a single customer had 60,000 desktop systems infected and over 500,000 e-mail copies in circulation within the company.

The Risks Digest [28] (comp.risks) noted that Microsoft blocked all outgoing e-mail to guard against propagation of the virus outside the company.

In its FAQ, [24] the Software Engineering Institute's (SEI's) Computer Emergency Response Team (CERT) noted first-hand reports from 300+ organizations that had been infected. Across these organizations, over 100,000 computers were infected. This infection spread rapidly. From first reported infection to over 100,000 infections took less than three days.

Remarks

Melissa, like the Internet worm, targeted very specific software. In the Melissa case, Microsoft Word versions 8 or 9 were the only software that could be infected. Systems that did not use this software could not be infected. Note, however, that Word version 8 was available for both the Microsoft Windows operating system and the Apple MacOS operating system so both systems could be infected.

Melissa used only the Microsoft Outlook e-mail client to propagate itself to other systems. Users of Microsoft Outlook Express, Netscape Mail, Eudora, or other e-mail clients could themselves be infected if they used MS Word, but would not automatically spread the virus. Of course, having been infected, any Word files e-mailed manually would spread the infection.

Further, while users of e-mail clients other than MS Outlook did not automatically propagate the virus, they were frequently victims of colleagues and acquaintances who did use Outlook and were flooded with e-mail sent by Melissa infections on other systems.

4. THE LOVELETTER WORM

The LoveLetter worm (VBS.LoveLetter.A) was released to the Internet in May 2000. LoveLetter is a Microsoft Visual Basic script (VBScript) worm that is delivered to victims as an e-mail attachment.

Techincal Summary

The LoveLetter worm takes advantage of the Windows Scripting Host (WSH) capability of Microsoft Outlook. When a victim clicks on an e-mail script attachment in Outlook, Outlook invokes WSH to execute the VBScript, which infects the victim's system. Other VBS-enabled e-mail clients can also execute the worm script. The worm is restricted to Microsoft environments because the Visual Basic programming language is only available from Microsoft.

The worm performs the following actions:

- Copies of the VBScript program are stored in several folders on the C: drive. The copies ensure that the worm is restarted after a re-boot.
- The Windows registry is modified so that the worm script file is invoked each time the system is restarted. This ensures that the worm is always running.
- The Windows Registry is modified to remove keys that disable password caching and that hide shared passwords.
- The Windows Registry is modified so that starting Microsoft Internet Explorer causes the download of a password-stealing Trojan program. This program sends stolen passwords to an e-mail address at system startup and at certain other times. The Windows Registry is modified to ensure that this Trojan runs at each re-boot of the system.
- A copy of the e-mail and infected script is sent to every entry in the Microsoft Outlook address book. Recipients who open the e-mail attachment become infected, thus spreading the virus.

Additionally, the volume of e-mail causes a significant increase in e-mail activity, impacting e-mail servers.

- A copy of the virus, encapsulated inside an HTML file, is sent to users who join IRC chat groups used by the victim.
- Files with certain file extensions are deleted, and a copy of the worm is stored using the name of the deleted file combined with a new extension .vbs. If a user clicks on this new (but familiar-looking) file name, the worm is re-executed.

The worm avoids casual detection by taking advantage of common Microsoft conventions:

- Installation of Microsoft Internet Explorer also installs WSH by default. This also links WSH to Outlook, such that clicking on an e-mail attachment automatically launches WSH to execute the script. While many users had installed Microsoft Internet Explorer, few realized that this installation gave Outlook the capability to execute scripts via WSH.
- A common Windows default is to suppress the display of file extensions. The e-mail attachment is named LOVE-LETTER-FOR-YOU.TXT.vbs. If file extensions are suppressed, the user sees a file name of LOVE-LETTER-FOR-YOU.TXT. Users may assume that the attachment is a text file with no associated application, and assume that it is "safe" to open the attachment. However, rather than displaying a text file, the VBScript file is executed and the victim is infected. The virus also replaces certain files on the user's hard drive with similarly named versions of the virus. Thus clicking on files with certain extensions (.jpeg or .mp3, for example) will re-launch the virus.

Copies of the virus and associated files are stored with "Windows-like" file names: MSKernal32.vbs, Win32DLL.vbs, and WIN-BUGSFIX.EXE. It is difficult for an average user to recognize whether a "Windows-sounding" file name is legitimate or not.

Extent of Infection

It is difficult to assess the overall extent of this infection. Symantec reports the worm "has wide-spread distribution, infecting millions of computers." [12] Such numbers are, at best, merely guesses. With many organizations reluctant or unwilling to provide accurate numbers of systems infected, the extent of such widespread infections will never be accurately known.

The worm causes much damage when a system is infected:

- Files are destroyed
- Passwords are stolen
- E-mail servers are clogged by copies of the worm

Additionally, it takes time to recover from infection of a host. Even uninfected users typically had to spend time dealing with the worm, either deleting the e-mail sent by infected hosts or updating anti-virus software to prevent infection.

Variants of the LoveLetter worm were easily created and re-introduced. Early variants had a different subject for the e-mail or different text in the e-mail body. (One version purported to be from Symantec Anti-Virus Research Center containing a file to protect against the worm: the file was the worm itself.) Other versions changed the processing of the worm. As of August 2000, there were 29 reported variants of the LoveLetter worm [12], and detection systems continue to report interception of the worm.

Remarks

The LoveLetter worm is written to exploit both the human "weak-link" and vulnerabilities in the Windows environment. It illustrates how easily a homogeneous environment can be exploited. The worm is written in a Microsoft-specific language, is launched (with a user action) from any e-mail application which supports Microsoft WSH, installs itself on the system using the Microsoft Registry, and spreads itself using the Microsoft Windows Address Book facility. With

Microsoft products installed on the vast majority of end-user systems, it is easy to exploit Microsoft vulnerabilities (and normal capabilities) to have a significant impact on users.

However, running a non-Microsoft environment (and thus avoiding Microsoft-only programs) is not free of impact:

- The ILOVEYOU worm flooded e-mail systems. This impacted many servers, as they crashed or were taken offline for repair / protection. This impacted all e-mail users, whether running the Outlook client or not.
- The script was written in VBScript, an ActiveX scripting language. ActiveX can host many scripting languages, including Perl and TCL/TK. Scripts written in these languages have the potential to run on systems other than Microsoft. Using those languages, it would be easy to write a script that could be destructive in additional environments. One variant of the LoveLetter worm is a version written as a generic UNIX shell script.

Other mail clients, such as Netscape Communicator and Eudora, can launch VBScript attachments to e-mail on a Windows platform via WSH. Thus, even different e-mail applications are exposed to vulnerabilities in common facilities.

5. DISTRIBUTED DENIAL OF SERVICE ATTACKS

During February 2000, several high-profile Internet sites were crippled by Distributed Denial of Service (DDoS) attacks. During a 3-day period, Yahoo, Buy.com, eBay, CNN, Amazon.com, ZDNet, and others were flooded with network traffic, either crashing servers or rendering them inaccessible to users. A 16-year old youth is accused of launching the attacks from his home.

Technical Summary

The DDoS attacks were launched from a network of compromised systems running the attack software. Using this hierarchical network of "slave" and "master" systems, a single attacker is able to mount a massive attack against a victim on a scale that overwhelms it. With a large number of attacking machines, an attacker does not need to exploit vulnerabilities in the victim: the victim can be overwhelmed with an extremely large flood of valid transactions.

The tool used to execute the February DDoS attacks is reported to be the Tribal Flood Network (TFN), which runs on UNIX systems. To install TFN, the UNIX host must first be compromised by exploiting a vulnerability. Once compromised, the host is modified to install both the TFN tool and a "root kit," which helps prevent the detection of tools such as TFN. This compromised host becomes a "slave" or "master" in the attack network hierarchy, and is ready to be used in a DDoS attack.

When an attack is launched, the attacker selects a victim, generally by specifying an IP address. The master systems each direct several slave systems to begin the actual attack. Several attack methods are available, using different methods to flood the victim. The TFN tool can attack with either a UDP flood, a TCP SYN flood, an ICMP flood, or a smurf attack [3].

Extent of Infection

There are two groups of systems affected by DDoS attacks: victims and compromised hosts.

Victims are selected by the attacker. Therefore, the attacker decides the extent of an attack. If the attack is directed against a server, other systems are affected indirectly, as they cannot access the server's services. For these DDoS attacks, all victims are attached to the Internet, and are important sites on the Web. Every system connected to the Internet is a potential victim of a DDoS attack, so the potential extent of impact is enormous.

Compromised hosts are used as slave or master hosts by the attacker. The TFN tool used in the attacks runs on UNIX systems, primarily Sun Solaris and Linux. Recently, some DDoS programs have been ported to the Windows environment, but UNIX machines are most often used for DDoS attacks. In order to mount a strong attack, a network of compromised hosts is created. These networks are often large. The February DDoS attack network is reported to consist of at least 50 computers. In a report on the trinoo DDoS tool [23], some attack networks consisted of 227, 888, and 10549 compromised hosts.

It appears to be easy to create a network of compromised systems. Attackers can scan systems connected to the Internet, obtaining information about the operating system level in use. With this information, the attacker selects a tool that can compromise the system. Many of these tools are collected into toolkits and are readily available on the Web. These toolkits effectively "automate" the process of finding and compromising systems.

The I-CAT Metabase [10] has been collecting profile statistics about vulnerabilities reported by CVE. These vulnerabilities are categorized by the targeted operating system. For 1999, 286 new vulnerabilities showed the following distribution of target systems:

- UNIX 51%
- Windows 95 family 23%
- Windows NT family 37%

For the year 2000, the distribution of vulnerabilities is similar. There is a continuous stream of new vulnerabilities found and available for compromising the systems in use today.

Remarks

DDoS attacks overwhelm the victim by sending far more IP transactions at one time than it can handle. Although other attack methods may crash the victim by exploiting flaws in the software, it is equally effective to overwhelm the victim with a massive amount of legitimate traffic, leaving the victim unable to process other requests.

There are few effective methods for dealing with these attacks. If the victim is disconnected from the network in order to protect it, the attacker has succeeded in removing the victim from normal service. In some cases, the attack network traffic can be re-directed by changes in up-stream components such as routers, but it takes time to determine the source of the attack and implement the configuration changes.

The same TCP/IP connectivity that enables the Web has become a common point of vulnerability for attackers to exploit. Even entirely different systems are vulnerable to attacks on the common components. Each implementation of the common facility has potential vulnerabilities that can be explored with a common set of approaches and tools. Additionally, entirely different implementations of a common facility will fail when the attack exploits the normal functions of the common facility.

6. DISCUSSION

At best, the attacks presented here are "anecdotal" evidence that diversity improves survivability. This is because many of the attacks are targeted for one system: the non-targeted systems are not affected. The available data on incidents is not complete enough to form the basis of a conclusion. However, there is no evidence in any of the examples presented that suggests diversity reduces survivability.

DesWarte *et al.* [6] describe many different examples of the use of diversity in industrial software engineering and in business practices. For example:

- Airbus A-300/600 digital fly-by-wire system is run by two classes of computers with different microprocessors designed independently and provided by different vendors.
- Boeing uses two different compilers to compile separate instances of the fly-by-wire software on the 777 aircraft.
- Separation of duties, a common business practice to prevent/deter fraud is a form of diversity.
- Software testing uses multiple approaches (e.g., code reviews, functional tests, code coverage testing) because each approach

is likely to find different types of problems. Therefore, collectively these approaches produce higher software quality.

Essentially, they argue that diversity must be good since it is used in many different ways to provide security and reliability.

Most of the Morris worm's chroniclers took no position on the issue of the advantages or disadvantages of a homogeneous versus heterogeneous networking environment with respect to information system survivability. However, Eichen and Rochlis did make the point at the time that [8]: "Diversity is good. Though the virus picked on the most widespread operating system used on the Internet and on the two most popular machine types, most of the machines on the network were never in danger. A wider variety of implementations are probably good, not bad. There is a direct analogy with biological genetic diversity to be made."

The examples do support a need to better understand the role diversity plays in survivability and defense. For example, it was noted that the Morris Worm could only propagate over TCP/IP connections. Potentially vulnerable systems (those running BSD 4.2 or 4.3 UNIX derivatives) were not affected if they were connected via UUCP or X.25. In the more recent Melissa case, network connection protocol was not considered because all systems used TCP/IP. Connection protocol diversity has been dramatically reduced in recent years with the arrival of TCP/IP for virtually all commonly used hardware and software.

Along these lines, the recent denial of service attacks against Yahoo and other sites suggest that use of a single common communication protocol makes everyone vulnerable to the same attacks. Hence, diversity, like other protection mechanisms is likely to require layering or "Diversity in Depth" to provide good protection. Diverse service implementations that rely on a common communication mechanism will not survive attacks on the shared mechanism. This is another example of common mode failures interfering with planned diversity. In other words, a homogeneous information system consists of a logical single point of failure.

Another question that needs consideration is the level of diversity required to derive significant benefit. In the examples presented, the level of diversity was relatively low, a few different implementations to perhaps tens of implementations in the case of UNIX variants. Is

this enough to "raise the bar" for a determined adversary or is diversity on a large scale necessary (as in hundreds or thousands of variants)? For example, automatic software mutation as described in Michael *et al.* [15] can make each running copy of a program unique. Other approaches to building diverse computer systems [9] could be effective for attack techniques known today (e.g., the buffer overflow) and suggest other methods for protecting system data (e.g., unique, changeable signatures for files.)

However, it is unknown at this time whether these methods are effective or practical in creating an environment of diversity. Some research indicates that it may be difficult to "create" diversity by modification of software:

- The authors in Michael *et al.* encountered several problems with the software mutation approach. They report "… doubts as to whether source-code mutation is a viable way to create software diversity." Additionally, their work with abstract interpretation of a program resulted in too many constraints to be processed by the system. This lead them to believe that only random constraints (i.e., a subset of the total number of constraints) could be effectively monitored to detect software modification [15].

- In an e-mail note to the BUGTRAQ list, Crispin Cowan of the Oregon Graduate Institute states "… we investigated the approach of using diversity to resist attack, and found it to be VERY limited in effectiveness" [1]. This was because the things that must be preserved (for a program to work properly) and those that must be changed (to ward off an attack) are largely unknown. Their research efforts turned to "restrictions," which essentially wrap additional checking around critical components [2].

One thing that seems obvious from both the Morris worm and Melissa discussions is that the authors could have made these attacks capable of handling more systems. In the Melissa case, using the Mail Applications Programming Interface (MAPI) instead of spawning Outlook would likely have enabled Netscape and Eudora e-mail clients to spread the infection automatically. The Morris worm had capabilities that were not used that could have supported additional UNIX platforms. Additionally, attack kits combine several attack tools and provide easy selection of the tool which can exploit the target

system vulnerabilities. Hence, a determined adversary might easily defeat small-scale diversity.

It may also be the case that extensive diversity creates additional areas of exploitation. Where implementations are different, the possibility exists for new errors caused by these differences. These errors could potentially be exploited for attack, resulting in vulnerabilities that did not exist in the homogeneous environment[1].

7. CONCLUSION

The attacks studied here illustrate that diversity in computer systems appears to be desirable: the specific systems / facilities not targeted do survive an attack. In an environment with different implementations of services, this diversity helps create forms of redundancy: some implementations continue to operate when others have failed. This creates a greater level of system availability and reliability, and as a result there is an improved confidence in the integrity of the information system.

However, the computer industry is moving toward a more homogeneous environment. There has been a steady consolidation of operating systems and applications, despite a tremendous growth in the number of users. Additionally, there is increased popularity in common services based on standards (e.g., TCP/IP) or open (i.e., shared) software (e.g., Linux). This homogeneous environment remains highly vulnerable to attack.

It also appears that diversity may be difficult to "create" in a homogeneous environment. Several researchers have reported complex problems in attempting to modify software to introduce immunity to certain attacks. The large number of attacks discovered each year implies that new "diversity" methods will constantly need to be created for effective defense. It remains to be seen whether

[1] This opportunity to exploit inconsistencies among multiple implementations was suggested by Julie Bouchard of Sandia National Labs during planning for the DARPA Information Assurance program's RT 0001 exercise.

practical methods will be created to provide sufficient diversity to help defend against attacks.

BIBLIOGRAPHY

1. Cowan, Crispin, e-mail subject ”Diversity (was: IIS Remote Exploit (injection code)),” BUGTRAQ mailing list, June 16, 1999.
2. Cowan, Crispin and Pu, Calton, “Death, Taxes, and Imperfect Software: Surviving the Inevitable,” http://www.cse.ogi.edu/DISC/projects/immunix/publications.html; presented *New Security Paradigms Workshop 1998.*
3. Criscuolo, Paul J., “Distributed Denial of Service,” CIAC-2319, Department of Energy Computer Incident Advisory Capability, February 14, 2000.
4. Denning, Dorothy, *Information Warfare and Security*, Addison-Wesley, Reading, 1999.
5. Denning, Peter J., "The Internet Worm," in Peter J. Denning, ed., *Computers Under Attack: Intruders, Worms, and Viruses*, ACM Press, N.Y., 1990.
6. DesWarte, Kanoun, and Laprie, "Diversity against Accidental and Deliberate Faults," Computer Security, Dependability, & Assurance: From Needs to Solutions, IEEE Press, 1998.
7. Dittrich, David, “The DoS Project’s “trinoo” distributed denial of service attack tool," University of Washington; http://staff.washington.edu/dittrich/misc/trinoo.analysis.
8. Eichin, Mark W. and Rochlis, Jon A., "With Microscope and Tweezers: An Analysis of the Internet Virus of November 1988," Massachusetts Institute of Technology, Cambridge, 1988.
9. Forrest, Spephanie, Somayaji, Anil, and Ackley, David H., “Building Diverse Computer Systems,” *Proceedings of the 6th*

Workshop on Hot Topics in Operating Systems, IEEE Computer Society Press, Los Alamitos, CA., pp. 67-72 (1997).

10. http://csrc.nist.gov/icat/
11. http://www.attrition.org/mirror/attrition/stats.html
12. http://www.symantec.com/avcenter/venc/data/vbs.loveletter.a.html
13. Kahney, Leander and Polly Sprenger , "Melissa, Spawned by Spam," Wired News, http://www.wired.com/news/news/technology/story/18819.html
14. Lottor, Mark, "Internet Growth (1981-1991)," RFC 1296, Network Working Group, January 1992.
15. Michael, C.C., Aron Bartle, John Viega, Alexandre Hulot, Natasha Jarymowyzc, J. R. Mills, Brian Sohr, Brad Arkin, "Two Systems for Automatic Software Diversification," DISCEX, 2000.
16. Montz, Lynn B., "The Worm Case: From Indictment to Verdict," in Peter J. Denning, ed., *Computers Under Attack: Intruders, Worms, and Viruses*, ACM Press, N.Y., 1990.
17. Ohlson, Kathleen, "Melissa: The Day After," Computerworld Online News, 30 March 1999.
18. Page, Bob, "A Report on the Internet Worm," Computer Science Department, University of Lowell, November 7, 1988.
19. Reynolds, Joyce K., "The Helminthiasis of the Internet," RFC 1135, Network Working Group, December 1989.
20. Rochlis, Jon A. and Eichin, Mark W., "With Microscope and Tweezers: The Worm from MIT's Perspective," in Peter J. Denning, ed., *Computers Under Attack: Intruders, Worms, and Viruses*, ACM Press, N.Y., 1990.
21. Seely, Don, "A Tour of the Worm," Department of Computer Science, University of Utah, n.d.
22. Shoch, John F. and Hupp, Jon A., "The 'Worm' Programs - Early Experience with a Distributed Computation," in Peter J.

Denning, ed., *Computers Under Attack: Intruders, Worms, and Viruses*, ACM Press, N.Y., 1990.

23. Slade, Rob, "Melissa Macro Virus," The Risks Digest Vol 20, Issue 26, 1 April 1999.

24. Software Engineering Institute (SEI) Computer Emergency Response Team (CERT), "Melissa FAQ," http://www.cert.org/tech_tips/Melissa_FAQ.html

25. Spafford, Eugene H., "The Internet Worm Incident," Technical Report CSD-TR-933, Department of Computer Sciences, Purdue University, West Lafayette, September 19, 1991.

26. Spafford, Eugene H., "The Internet Worm Program: An Analysis," Purdue Technical Report CSD-TR-823, Department of Computer Sciences, Purdue University, West Lafayette, November 29, 1988, revised December 8, 1988.

27. United States General Accounting Office, "Computer Security: Virus Highlights Need for Improved Internet Management," GAO/IMTEC-89-57, United States General Accounting Office, June 1989.

28. Woods, Lloyd, The Risks Digest Vol 20, Issue 26, 1 April 1999.