INVITED TALK 5

J. H. P. Eloff et al. (eds.), *Information Security — the Next Decade*

37

Viruses, Corruption, Denial, Disruption, and Information Assurance

by Dr. Frederick B. Cohen
Management Analytics
PO Box 1480
Hudson, OH 44236
USA
Tel:US+216-686-0090
Fax:US+216-686-0092

Abstract

In this paper, we begin by describing the historical roots of widespread corruption of information and denial of services paying particular attention to the increasingly networked information environment. Next we describe the historical emphasis on secrecy, fault tolerant computing, and safety, and show how these fields of study do not provide adequate coverage against intentional denial of services or corruption of information. We then define the terms disruption and information assurance, discuss some of the elements that this subfield should encompass, summarize some of the current work in this field, and suggest future directions.

1 Background

Most of the world now depends on information technology in one form or another for everything from controlling electrical power grids to keeping track of bank accounts. Our communications systems, transportation systems, governmental systems, business systems, and even our cars and gasoline pumps depend on information technology for their proper operation. Furthermore, we are increasingly networking our computers together. The Internet, as an example, now directly connects about 2.5 million computers, and indirectly connects something like 250 million computers. With increasing connectivity comes increasing risk, and as we will now see, the networked environment has become a very risky place to operate.

In the 1960s, sporadic reports of positive feedback loops in network protocols began to appear as the early ARPAnet computer network began to take shape. At about the same time, science fiction began to discuss the concept of live computer programs when "The Shockwave Rider" [1] described a program that moved through the computer networks of the world gathering information. But it wasn't until 1982 that the first journal paper came out on the Xerox "worm" experiments designed to explore parallel processing on a distributed computing network. [2] In 1984, the first paper was published on computer viruses [3], and the realistic possibility of widespread intentional corruption of information residing in modern information systems and networks became clear.

The first widely publicized global example of a virus in an information network came in 1988, when IBM mainframes around the world were infected by the Christma.Exe virus [4], and only a year later, service on the Internet's then 60,000 computers were disrupted by the Internet Virus. [5] [6] Since that time, the number of computer viruses has grown to over 5,000, with a new one now being created about every 8 hours. The incidence of virus-induced disruption has become so high that most substantial businesses expect several incidents per year.

Despite their high profile and high incidence rates, computer viruses are not the only cause of widespread service disruption or information corruption in information systems today. One recent survey of denial of services identified about four billion dollars of losses in 1992 alone, [7] and AT&T reports figures of 2 billion dollars per year in toll fraud losses, much of it due to the corruption of PBX systems. Several authors have reported that once detection was put in place, over one attack per day was detected against their computers attached to the Internet. [8] [9] [10] Other people have placed "honey pot" systems on the Internet to detect attacks and have privately reported similar figures. There are about 2.5 million computers on the Internet, so simple multiplication tells us that something like 900 million attacks per year take place in that venue alone.

There are now automated tools in use on the Internet that probe for new systems, attempt to enter them, and plant subtle entry points for future use. [13] In many experiments, every

host attacked has been taken over. For example, a widely published attack breaks into a significant percentage of all of the computer in the Internet by simply sending electronic mail. In one case, a Trojan horse was planted in the "Internet Relay Chat" (IRC) source code and widely distributed throughout the Internet. [14] In another case, many of the key routing computers used in the Internet were corrupted to gather passwords sent between sites. This attack is still underway after more than a year, nobody has been caught, and passwords to more than 100,000 system have now been gathered. [15] Industry sources have confirmed that over 80 percent of attempted attacks against customer computers attached to the Internet succeed.

Electronic mail can be used to flood computer systems with information, thus reducing available resources for other purposes. In experimental attacks, this technique was shown to be very effective in denying services, [11] and several incidents since that time have demonstrated the same results with accidental causes. [12] In one experiment against military systems, 56 out of 68 sites were found to be vulnerable to this attack, and the Internet now loses large parts of its mail capabilities in the United States about once per month for a period ranging from hours to days from apparently accidental causes. The vast majority of modern electronic mail systems allow forgeries to be made very simply, and many organizations use electronic mail as a primary means of inter-office communication.

These examples are only the tip of the proverbial iceberg. Many more examples have been cited in the literature [16], and I urge anyone who has found these examples interesting to take the time to thoroughly research this subject.

2 Some History of Information Protection

During the same span of time that much of modern society became highly dependent on information technology and the number, likelihood, and severity of incidents of corruption of information and denial of services rose, the field of information protection was being formed and expanded. Unfortunately, historical circumstances combined to cause a gaping hole in the development of information protection. [16]

One important factor in the development of what I call information protection was its separation into subfields, typically called *cryptography*, *computer security*, *fault tolerant computing*, and *software safety*, in order of their emergence.

By its very name, computer security reflects its association with police forces and military organizations. The concept of information theft was easy to grasp, people in the security field could easily relate to theft of information, the legal system had historical provisions for dealing with thieves, and the militaries of the world had thousands of years of history associated with information secrecy. Some of the earliest mathematical developments included

the Bell-Lapadula model based on military secrecy requirements [17], the Harrison, Ruzzo, Ullman model, which was interpreted primarily in terms of the ability of subjects to read objects, [18] and the lattice model, which was considered primarily in terms of its ability to model compartments as well as security levels. [19]

Even though the latter two of these models are very general purpose, their interpretation has been largely in terms of their secrecy implications. This is more of a reflection on the field of computer security being highly context bound than a reflection on the ability of the models to be used for other purposes. For example, in 1986, these two models were used to show that computer viruses could not be prevented in a system allowing sharing, transitive information flow, and programming [20] and to show how limited information flow could be effectively used to limit the spread of viruses. [21] [22] Nevertheless, the field of computer security is still largely concerned with protecting secrets.

A very similar situation occurred in the area of cryptography. For about 5,000 years, cryptography concentrated on keeping secrets. [23] It has only been a few years since substantial use of cryptographic checksums have been employed for protecting integrity. This effort started in ernest in the banking industry, where electronic funds transfers came to be so vital and subject to abuse that a means had to be found to assure that transmissions were not altered or forged. Many schemes were created in the 1960s and 1970s to address this problem, and several were adopted in the banking industry.

The next substantial breakthrough in cryptography for protection of integrity occurred with the creation of the concept of public key encryption [24] and the subsequent development of the most commonly used and probably most mathematically tested public key system, the RSA. [25] These cryptographic systems can be used to sign documents, to create very hard to forge cryptographic checksums, and for numerous other purposes. Unfortunately, they are somewhat slow to use, and have only come to be used by a relatively small number of people with special requirements. They are far from universally applied, even though they are inexpensive and highly effective against many sorts of corruption.

Perhaps the reason that high integrity cryptographic checksums are not widely used today can be traced back to the development of the field of fault tolerant computing. Beginning with the early work of John von Neuman, [26] information science has concerned itself with issues of reliability in the presence of random faults. Much of the development of information theory is based on Shannon's work in 1948 [27] and this theory is widely used in fault tolerant computing and various parts of coding theory. Yet the very important theory of secrecy systems also published by Shannon only a year later [28] is widely unknown in the information science community. But this theoretical issue is only one of many challenges left unmet.

For example, existing fault tolerant computing standards and practices such as protection against random noise, [29] [30] lightning, [31] RF noise, and [32] loss of packets, [33] only address random stochastic noise. Unfortunately, intentional attackers are not accurately modeled by the statistical models of faults used to develop existing reliability standards.

Let's look at what a typical text says on the subject: "The noise analysis of communications systems is customarily based on an idealized form of noise called 'white noise', the power spectral density of which is independent of the operating frequency. The adjective 'white' is used in the sense that white light contains equal amounts of all frequencies within the visible band of electromagnetic radiation. ..." [35] The reason cited for the random noise models is the ease of analysis, [36] but ease and adequacy of analysis are not always compatible.

One of the most common techniques for detecting corruption in memory and transmission is the use of a 'parity' bit associated with each byte. The parity bit is set to 1 or 0 to make the total number of '1's even (or odd, depending on whether the even or odd parity convention is being used). This technique detects ALL single bit errors, which is quite effective against particular sorts of random noise that cause transient faults. It is not effective against an intentional attacker who can change sets of bits collectively while maintaining parity, thus keeping the parity the same while corrupting the information and avoiding detection.

On disk storage, in LAN packets, and in some satellite transmission, CRC (Cyclical Redundancy Check) codes are used to detect classes of faults that result in errors to linear sequences of bits of at most some predefined length. [33] Again, these codes are ineffective against an intentional attacker, because it is easy to determine the constant coefficients of the coding equations by watching packets, and from this it is easy to forge packets at will undetected. [4]

Current system reliability estimates do not account for deliberate software corruption. [34](p 55) Telecommunication networks can fail from software malfunction, failures can propagate in operations or control systems, [37](p32) and system availability estimates seem to overlook this cascading effect. As an example, telephone networks are supposedly designed for something like 5 minutes of downtime per year, [38] and one company advertises that if 800 service fails, restoration is guaranteed in under 1 hour. Yet in a single incident in 1990, the AT&T (American Telephone and Telegraph) 800 network was unavailable for over 4 hours, [37] which seems to imply that this failure covers expected outages over the next 50 years! Considering that a similar failure brought down telephones in several major cities for several days in 1991, [39] there appears to have been a flaw in this availability analysis.

Current public telecommunications networks lack adequate service assurance features to withstand intentional attack. Service assurance features are designed into these systems at every level, [37] and yet they still fail to meet even the challenge of accidental errors and omissions. As an example, in 1991, there was a major failure in telephone switches in several large US cities that lasted for several days, and was finally traced to a 3 bit error (a 'D' instead of a '6') in one byte of a software upgrade. [39] This is the simple sort of mistake that even minimal software change control detects. This change was apparently never tested at all, was put into widespread use, and caused widespread harm. In 1990, AT&T's long distance network shut down due to a protocol error that impacted millions of customers nationwide for over 4 hours. [40]

In many cases, telecommunications disruptions must be resolved in very short time frames. For example, some telephone switching systems must be repaired within 1.5 seconds or the circuit failure errors passing through the network will cause a propagating positive feedback which may deadlock more of the network, [41] eventually cascading into a major problem. An attacker only needs to disrupt two sites for 1.5 seconds to cause such a cascading effect.

One quarterly report of large scale disruption incidents for the fall of 1993 includes an FAA (Federal Aviation Administration) computer systems failure delaying regional traffic for 90 minutes (cause unknown), an FAA weather computer system failure for 12 hours due to a time activated logic bomb, a programming error in an X-ray system that resulted in wrong dosages to about 1,045 cancer patients, and a software 'bug' that crashed the Hamburg ISDN (Integrated Services Digital Network) telecommunications services for over 11 hours, and this is only one of several similar publications that report different incidents. [42]

Similar lapses in policies and procedures seem to be common for major software manufacturers. As an example, in 1992, Novell released a virus to tens of thousands of customers when it was noticed **after quality control was completed** that a key file was missing from the master distribution disks then being transported to the disk duplication facility. Instead of returning to the quality control process, the person transporting the disks for duplication loaded the file from the most convenient computer, which by chance contained a virus that was transferred to the floppy disk. The disk was sent to duplication, packaged, and shipped to customers. [43] The disks were apparently never tested at random after duplication for problems, the disks en-route to the duplication facility were not sealed or permanently write protected, the personnel were not properly trained, and the initial quality control process never detected that the correct file was not on the disk!

According to the United States National Research Council, "Most communication channels incorporate some facilities designed to ensure availability, but most do so only under the assumptions of benign error, not in the context of malicious attack." [34](note 6, p100)

The field of 'high assurance' computing addresses information systems for the most critical applications. (e.g., life support systems, flight controls, nuclear warhead detonation) Unfortunately, building 'perfect' systems is far too costly and resource intensive for the wide variety of systems and networks found in today's environment, and only adequately addresses certain types of very well defined control applications. This work is oriented toward designing a perfect system wherein all inputs, states, outputs, and state transitions are specified in full detail, and mathematical proofs are provided to show that the design is properly implemented. [44] [45] Although this type of solution may be applicable to certain limited control problems in embedded systems, these sorts of solutions are computationally infeasible for any large system, only cover sufficiency and not necessity [20], only cover limited function systems against disruption, [4] and are beyond current and anticipated capabilities over the next 20 years for the sorts of systems used in modern networks.

An alternative path to a similar solution is the use of automated program generating

programs. In this technology, a small number of programs are designed to automatically write the rest of the programs. Designers spend a great deal of time and effort in perfecting the design automation system which, in turn, designs other systems. [46] This technology is far from perfected, and even if it were perfected, it leaves the problem of writing perfect specifications, which is at least as hard as writing a perfect programs.

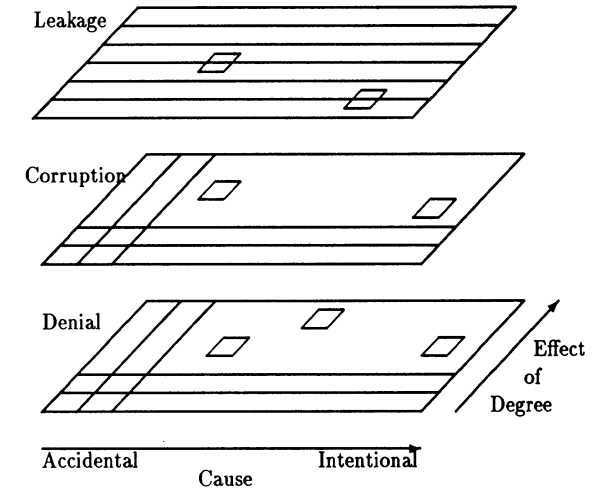
In the hardware realm, design automation has been highly successful, but this does not imply that it will be successful in the software realm. There are substantial differences between hardware and software. For example, the complexity of current software is many orders of magnitude higher than the most complex automated hardware design; the physical properties of hardware are abstracted out of most software design; software is designed based on a finite but unbounded randomly accessible space, while hardware is designed based on a relatively small finite and bounded space with only local access as provided by explicitly created wires. Furthermore, hardware design automation takes substantial amounts of computer time, still leaves design flaws such as data dependencies that have resulted in disruption, and is based on specifications that are susceptible to errors.

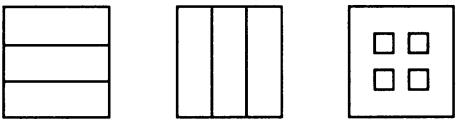
Another alternative is the use of extremely intensive testing to detect the presence of errors and correct them. The problem with this approach is that testing for 100 percent coverage is as complex as perfect design. Imperfect testing leaves systems that fail when 'rare' events occur. In one study, the combination of two events characterized as low probability caused 50 percent of systematically designed, well tested, small control programs to fail. [47] If this is the current state of the art for low probability events in small programs, extremes in testing are not likely to be successful against intentional attacks on large globally networked infrastructures. All an attacker has to do is create two "unlikely" events to attain a 50 percent success rate.

For the sorts of general purpose systems in modern computing, there are classes of attacks that can not be perfectly defended against. Two well known examples are computer viruses [20] and the exploitation of covert channels. [48] If defenders spend resources trying to implement perfect solutions to these problems, they will surely fail and go bankrupt in the process, but defenders can not simply ignore these and other similar problems, because they present a real and identifiable threat to modern information networks. Feasible solutions will not be perfect. Rather, they should responsibly trade cost with protection.

3 Information Assurance

If we plot the potential for corruption of information, denial of services, and leaking of secrets under random events thru malicious attack, and show what areas of the plot are covered by current technology, we get an interesting picture:





Computer Security

Fault Tollerant Computing

High Assurance Systems

The clear implication is that, while computer security has covered leakage to a large extent, fault tolerant computing has covered accidental events to a large extent, and special purpose systems have selectively covered otherwise uncovered areas, there is a large, perhaps even dominant area left uncovered by the current state-of-the-art.

In this paper, we define the term *disruption* to describe the union of corruption of information and denial of services, and the term *information assurance* to describe the covering of disruption by protective techniques.

It is our strongly supported contention that information assurance is not adequately addressed by current protective measures, that insufficient research and development has been performed to effectively provide technical information assurance at this time, and that information assurance will be a core issue in information protection for many years to come.

4 What Do We Do Now?

The problems of information assurance have been around for a long time, and yet there have been very few substantial developments in the defense against intentional disruption. The few developments we have seen have not been very successful in the computing market. Here are some good examples of how the computing industry has consistently failed to address the issues of information assurance.

- When computer viruses first became an issue, it took several years of fairly intensive research to develop the few sound techniques available today. [4] Yet the vast majority of the computing industry has consistently decided to use unsound techniques that are ineffective against serious attackers and less cost effective than sound integrity techniques. The result is that the systems that support security services on the Internet are themselves subject to corruption, have been consistently corrupted, and are exploited to attack other systems. What good is Kerberos authentication when the authentication server has been taken over by the attackers?

- Cryptographic checksums have been in use by the banking industry for many years, and yet they are almost never used in other modern information networks despite the fact that non-cryptographic checksums are widely used and cryptographic checksums can be used for the same cost and without any reduction in performance. It took the CMU CERT team over a year to decide to advise the use of a cryptographic checksum to assure integrity in Unix systems despite the fact that their slower non-cryptographic checksum solution had failed after only a few weeks. Even at that, they don't use it properly, and forgers have apparently succeeded in making alterations that the system does not detect.

- Computer scientists have developed extensive capabilities for proving that programs meet sufficiency conditions to assure that they produce correct results, and yet I am unaware of any development that solves the far more important and easier to solve problem of assuring that programs properly handle exception conditions on returns from function calls. Programs that have been mathematically proven correct still produce wrong results when there isn't enough disk space to store their temporary files. Even when such simple measures as array bounds checking are available, they are almost universally not included in production runs for a relatively small performance enhancement.

- Compression is becoming a very popular and widely used strategy for reducing effective storage and transmission costs, and yet encryption is almost never used for assuring privacy or integrity. It is possible to do a simple encryption at the same time as a compression for almost no computational cost, and with no substantial impact on the

user, and yet vendors consistently choose not to provide this simple addition to their product, even as an option, and few of customers use the capability even when it is available.

- Almost every operating system in existence today, even DOS, offers discretionary access control (DAC), and yet only few vendors offer any form of mandatory access control (MAC), and those that do charge a lot of money for it. This is incomprehensible once you realize that providing MAC is *easier and less expensive* than providing DAC. Adding MAC to a system with DAC can be done in as little as a few hours. I know this because I have done it under both Unix and DOS in this time frame. We see vendors and university researchers spend hundreds or even thousands of hours devising special versions of command interpreters and administrative programs to detect wrong protection settings (e.g., the setuid bit under Unix) when a simple MAC policy would prevent this from occurring and a simple addition to audit trails would allow it to be detected in real time.

In analyzing this situation, one can't help but come to believe that something is wrong with this picture. Why would the people who make information technology decisions intentionally ignore a class of problems that is growing to the point of making their industry less viable and why would they ignore the solutions to these problems if they were no more expensive or usable than their current solutions? The only answer is that they would not do so intentionally. The only logical conclusion seems to be that they do not ignore the issues intentionally. They do so either out of ignorance or because they perceive some sort of competitive advantage. The competitive issues are beyond the scope of this paper, so we will concentrate on the issue of ignorance. The issue of ignorance can and should be addressed in different ways for different communities. The following list of suggestions may be a good starting point in this effort:

- To the extent that the members of the information protection community are ignorant of the issues underlying information assurance, it is important that they educate themselves and each other. This presentation is a starting point, but we need to spend a lot more effort to understand and address this issue more fully.
- To the extent that the members of the research community have not followed this line of research as heavily as they have followed the privacy issue, they should seriously consider changing directions.
- People who might want to achieve level of information assurance have to get funding. Funding in this area is almost non-existent because the people who have the funds don't understand the issues. Therefore, we must educate decision makers about the nature and scope of the information assurance issue in order to get them to emphasize it.

- For many years, programmers have been improperly trained for the information assurance task. A massive retraining effort would be required to reverse this trend any time soon, but over a period of ten or more years, this can be accomplished by starting to embed the information assurance issue in our normal professional training programs.
- The universities of the world have, with a few notable exceptions, provided inadequate education in the information assurance area, and that, to large extent, is why we have the problems we encounter today. Programming is introduced as something that can be done very simply, but I am unaware of any university level course in which a student was expected to write a program that dealt properly with even relatively simple error conditions. In order for society to change over the long run, our educational system has to start to seriously address this issue.

The efforts of individuals will be very important, but organizations must also change if the information assurance issue is to be properly addressed. The lack of purely technical solutions is nothing new in information protection. In fact, there are few if any purely technical solutions in this field. Even the simplest technologies require human supervision and proper application in order to be effective. Furthermore, with the unsolvable nature of some important technical problems in information assurance, comes a need to have non-technical approaches.

Organizational approaches of various sorts have been used, and many papers have been written on the subject. The approach we have selected and used in many recent information assurance efforts is comprised a combination of protection management, protection policy, standards and procedures, documentation, protection audit, technical safeguards, incident response, testing, physical protection, personnel issues, legal considerations, protection awareness, training, and education. In addition, the organization must be suited to information assurance by being able to make the changes required in order to achieve adequate protection. [16]

5 What Do We Do Next?

This conference is about the next ten years of information protection, and for that reason, I find this topic a particularly good one. To understand why I think so, you only have to look at history. The computer virus issue first came to light in 1984 after about eight months of scientific work on the topic, and it took about ten years for the issue to become solidly embedded in the world view of information technology.

Similarly, the information assurance issue, as presented here, has only been considered in a scientific light for a very short period of time. This may be the first conference presentation

to look at these issues in this way, and there are certainly a lot of hard problems remaining to be solved. It is reasonable to believe that it will take the next decade before the information assurance issue is as thoroughly embedded in the world view of information technology as the computer virus issue is today.

Despite the loose similarity drawn between computer viruses and malicious disruption, a very important distinction should be drawn between them. Computer viruses can and do disrupt services and corrupt information, and they are predominantly used for malicious purposes. But as an area of study and concern, malicious computer viruses comprise a very small subpart of the overall problem of malicious disruption.

For that reason, it is highly unlikely that the solutions to malicious disruption will be anywhere near as well developed as the techniques we have available against computer viruses today. The generic information assurance measures that have proven effective against computer viruses have widespread application to other areas of information assurance, but such techniques as scanning for known attack patterns are far more limited in their utility against the broader problem of malicious disruption.

References

[1] "The Shockwave Rider"

[2] J. F. Shoch and J. A. Hupp, "The 'Worm' Programs - Early Experience with a Distributed Computation", CACM pp172-180, March, 1982.

[3] F. Cohen, "Computer Viruses - Theory and Experiments", IFIP-sec 84, also appearing in IFIP-TC11 "Computers and Security", V6(1987), pp22-35

[4] F. Cohen, "A Short Course on Computer Viruses", originally published in 1989 by ASP Press, now available in the second edition from John Wiley and Sons.

[5] E. Spafford, "Crisis and Aftermath", CACM, V32#6, June, 1989.

[6] J. Rochlis and M. Eichin, "With Microscope and Tweezers: The Worm from MIT's Perspective", CACM, V32#6, June, 1989.

[7] Melinda-Carol Ballou, "Survey Pegs Computer Downtime costs at $4 Billion", Computerworld, August 10, 1992, p53.

[8] S.M. Bellovin, "There Be Dragons", Proceedings of the Third Usenix UNIX Security Symposium, Baltimore, September, 1992.

[9] Bill Cheswick and S.M. Bellovin, "Firewalls and Internet Security - Repelling the Wiley Hacker", Addison-Wesley, June, 1994.

[10] Marcus J. Ranum, "A Network Firewall", Washington Open Systems Resource Center, Greenbelt, MD, June 12, 1992.

[11] F. Cohen, et. al., "Planning Considerations for Defensive Information Warfare - Information Assurance", Task Order 90-SAIC-019, DoD Contract No. DCA 100-90-C-0058, November, 1993.

[12] Wall Street Journal, "PSI tells BBS Advertiser to Stop", June 22, 1994 p B5.

[13] Dan Farmer and Wietse Venema, "Improving the Security of Your Site by Breaking Into It"

[14] CERT Advisory 94-14, "Trojan Horse in IRC Client for UNIX"

[15] CERT Advisory 94-01 "Ongoing Network Monitoring Attacks"

[16] F. Cohen, "Protection and Security on the Information Superhighway", John Wiley and Sons, 1995.

[17] D. E. Bell and L. J. LaPadula, "Secure Computer Systems: Mathematical Foundations and Model", The Mitre Corporation, 1973.

[18] M. Harrison, W. Ruzzo, and J. Ullman, "Protection in Operating Systems", CACM V19#8, Aug 1976, pp461-471.

[19] D.E. Denning, "The Lattice Model of Secure Information Flow", vol.19, no.5 p.236-43, May 1976.

[20] F. Cohen, "Computer Viruses", PhD Dissertation, University of Southern California, 1985, ASP Press.

[21] F. Cohen, "Protection and Administration of Information Networks Under Partial Orderings", IFIP-TC11 Computers and Security, V6(1987) pp118-128.

[22] F. Cohen, "Design and Administration of Distributed and Hierarchical Information Networks Under Partial Orderings", IFIP-TC11 Computers and Security, V6(1987)

[23] David Kahn, "The Codebreakers", Macmillan, 1967 p 266.

[24] W. Diffie and M. Hellman, "New Directions in Cryptography", IEEE Trans. on Inform. Theory, V-IT-22, Nov. 1976.

[25] R. L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," Comm. of the ACM, 1978, Feb Vol 21 #2, pp120-126.

[26] John von Neuman, "Design of Reliable Systems from Unreliable Organs"

[27] C. Shannon, "A Mathematical Theory of Communications", Bell Systems Technical Journal, vol.3 #27, July, 1948.

[28] C. Shannon, "Communications Theory of Secrecy Systems", Bell Systems Technical Journal, 1949, 656-715.

[29] "BOC Notes on the LEC Networks - 1990", SR-TSV-002275, Bellcore, Issue 1, March 1991

[30] NCSD 3-8, "Provisioning of Emergency Power in Support of NS/EP Telecommunications", April 2, 1991

[31] MIL-HDBK-419, "Grounding, Bonding, and Shielding for Electronic Equipment and Facilities", V2, Jan 21, 1982.

[32] RS-252-A, "Standard Microwave Transmission Systems", Electronic Industries Association, 1972

[33] ISO IS 8208, "Information Processing Systems - Data Communications - X.25 Packet Layer Protocol for Data Transmission"

[34] "Computers at Risk: Safe Computing in the Information Age", National Research Council, National Academy Press, 1991.

[35] S. Haykin, "Communication Systems", J. Wiley & Sons, Inc., 2nd ed., c1983.

[36] D. Minoli, "Cost Implications of Survivability of Terrestrial Networks Under Malicious Failure", IEEE Transactions on Communications, V28#9, September, 1980, pp1668-1674.

[37] W. Falconer, "Service Assurance in Modern Telecommunications Networks", IEEE Communications Magazine, June, 1990

[38] Jim Gray and Daniel P. Siewiorek, "High-Availability Computer Systems", IEEE Compter, September, 1991 pp39-48.

[39] "FCC Preliminary Report on Network Outages", Common Carrier Bureau, July, 1991

[40] M. Alexander, "Computing Infosecurity's Top 10 Events", Infosecurity News, September/October, 1993.

[41] Bob Pekarske, "Restoration in a flash - using DS3 cross-connects", Telephony, Sept 10, 1990.

[42] "EDPACS", Dec, 1993.

[43] R. Lefkon ed., "Data Processing Management Association Annual Computer Virus and Security Conference", New York, NY, 1992.

[44] "Safety Criteria and Model for Mission-Critical Embedded Software Systems", IEEE CH3033-8/91/0000-0069, 1991.

[45] H. O. Lubbes, "High Assurance Computing Software Technology Requirements", Naval Research Laboratory, 1991?

[46] Daniel P. Siewiorek, M. Y. Hsiao, David Rennels, James Gray, and Thomas Williams, "Ultradependable Architectures", Annual Review of Computer Science 1990 4:503-15.

[47] Herbert Hecht, "Rare Conditions - An Important Cause of Faults", IEEE 0-7803-1251-1/93, 1993.

[48] B. W. Lampson, "A Note on the Confinement Problem", Communications of the ACM V16(10) pp613-615, Oct, 1973.