Using Malware to Improve Software Quality and Security

John Aycock, Department of Computer Science
Stefania Bertazzon, Department of Geography
University of Calgary

About the Authors

John Aycock is an assistant professor at the University of Calgary in the Department of Computer Science. He received a B.Sc. from the University of Calgary in 1993, and an M.Sc. and Ph.D. from the University of Victoria in 1998 and 2001, respectively. His research interests include computer security, compilers and programming language implementation, compiler tools, system software, operating systems, and all things low-level. He conceived and taught the University's infamous "Computer Viruses and Malware" course.

Stefania Bertazzon is an assistant professor at the University of Calgary (Canada). Having completed a Masters in Economics and a Ph.D. in Geography, she is an economic geographer, focusing her research on the spatial aspects of economics, such as location, spatial interaction, and diffusion. She tackles economic problems with the tools of GIS (geographic information systems) and spatial statistics. Her interests include tourism, marine traffic, and a body of research in medical geography, specifically on the analysis of the socio-economic determinants of health and accessibility to heath care facilities in Canada.

Mailing Address: Department of Computer Science, University of Calgary, 2500 University Drive N.W., Calgary, Alberta, Canada T2N 1N4; Phone: +1 403 210 9409; E-mail: aycock@cpsc.ucalgary.ca.

Keywords

Software quality, software security, malware, ethics, economics

Using Malware to Improve Software Quality and Security

Abstract

Software vendors have very few incentives, and even have disincentives, to producing secure, high-quality software. "I4NI systems" are a new type of malicious software whose payload bears ill intent, yet would be voluntarily, knowingly installed by software vendors. Use of these systems would give vendors competitive advantages, as well as empower consumers with a direct way to influence a vendor's software quality. We give a detailed description of I4NI systems, along with extensive technical, ethical, and economic analyses.

Introduction

Historically, in small communities, reputation is paramount. There are ramifications for a craftsperson with shoddy work.

No such problems face people who craft and sell software.[1] There are end-user license agreements and disclaimers galore to hide behind, to begin with. These legal terms are not limited to commercial software, either:

> 'EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.'
> (Free Software Foundation, 1991, clause 11)

It can be argued that these agreements appear simply to protect vendors from litigation, but in fact they provide no incentive to produce high-quality, secure software. There are even occasional provisions in law to excuse vendor negligence:

> 'No action may be brought under this subsection for the negligent design or manufacture of computer hardware, computer software, or firmware.'
> (Fraud and Related Activity in Connection with Computers, 1996, section (g))

There have been attempts to manufacture ramifications for vendors, such as a California lawsuit against Microsoft for security-flawed software (Lohr, 2003). Still, in spite of this, 'the vendor is the ultimate blameless party' (Pfleeger, 2004).

Software defects have an additional side effect for commercial software vendors: an additional revenue stream (Barnes, 2004). Vendors who charge their consumers for upgrades are in part

---

[1]We will collectively call people who develop and sell software "vendors" in this paper; their customers will be called "consumers."

collecting money for fixing bugs in the original product. This business model assuages extreme fluctuations in product sales by encouraging consumers to buy upgrades in between major software releases. There are other advantages to the vendor, too, including curbing software piracy and modifying software licensing terms (Barnes, 2004). Seen in this light, there is very little motivation for vendors to produce good software.

How can this problem be addressed? In the remainder of this paper, we look at the problems facing software development, and present a novel solution to the problem which employs malware to provide quality incentives to vendors. We analyze our malware-based approach in technical, ethical, and economic perspectives, and finish by surveying other legitimate markets for malware.

Software Development

Software development is hard. Developing correct software is hard, developing secure software is hard, developing quality software is hard. But all these attributes - correctness, security, quality - are desirable from the consumer's point of view.

Quality software clearly must be correct software, software that does what it's supposed to. A concrete definition of what constitutes correctness can be captured through formal specifications, but use of formal specifications is more the exception than the rule.

It is becoming increasingly apparent that quality software must also be secure software. Secure software is more difficult to produce than correct software, though. Correct software does what it's supposed to; secure software does what it's supposed to *and nothing else*. For example, a program which performs its specified function without error may be correct, but if an attacker can induce a buffer overflow and cause the program to execute some shellcode, then the program is insecure; the program allows behavior it wasn't supposed to. (Arguably the distinction between correctness and security is artificial, because secure behavior could be specified as part of correct behavior. This still means that secure software is harder than correct software, though.)

Furthermore, security, the hardest part of software development, cannot be procrastinated. Security must be considered throughout the software development process (Wang & Wang, 2003).

We have not considered one important factor. Software development is fundamentally a human endeavor and is subject to error: design errors, implementation errors. Bugs. Any sizable piece of code is expected to have a certain number of bugs (Scheider, 1988), and it has become normal to expect buggy software (Cowan & Pu, 1998). Inevitably, some of these bugs will be exploitable as security holes.

Unfortunately, in the last section we saw that software vendors have little incentive, and substantial disincentives, to try and produce quality software with few defects. Even if the defects aren't fixed, a giant leap forward would simply be to know when problems are present, perhaps through testing done by an independent standards organization or government body (DePompa, 2004).

Liability for defective software could be imposed. This could obviously be done at the vendor level, but it could also appear at the consumer level, making consumers liable for using bad software (Fisk, 2002; Schneier, 2004). Either scenario would open the floodgates for insurance companies to insure against liability, and to reward less-risky software choices (Schneier, 2001). As part of their risk management, insurers would also pressure vendors to produce better software (Pfleeger, 2004).

I4NI Systems

There is another way to provide incentives to software vendors. It is complementary to testing and insurance, and can be viewed as a very direct, democratic means of assessing liability. It involves malware.

A vendor with faulty software can cause untold grief for their consumers, but there has been no direct way for consumers to share that pain with the vendor... until now. An "I4NI" system[2] is an arrangement between a vendor and their consumers, where the vendor voluntarily places malware on their computers; this malware is normally dormant, but may be activated by dissatisfied consumers to cause a negative impact on the vendor's computers.

There are several critical points to note.

1. I4NI systems are not cases of extralegal, vigilante justice; in fact, use of such a system may be contractually specified. A software vendor will have voluntarily entered into this arrangement, for reasons we will discuss in the next section.

2. Non-propagating malware only is used in an I4NI system, effectively logic bombs. Propagating malware - viruses and worms - have from earliest experiments shown themselves to move surprisingly quickly (Cohen, 1987) and be hard to control (Shoch & Hupp, 1982). Using viruses and worms would present far too high a risk of escaping onto machines that do not belong to the vendor and, consequently, the I4NI arrangement.

3. Checks and balances must be present. Obviously, a single disgruntled consumer should not be able to activate an I4NI system by themselves. A vote would have to be held amongst consumers, and both a quorum of consumers as well as a minimum "activation threshold" of votes would need to be met. A vendor in danger of having their I4NI system activated must also be given a reasonable time to try and respond to problems.

4. An I4NI system need not involve all of the vendor's computers.

There are several forms that an I4NI system might take. The obvious manifestation would be a system that, when activated, would simply destroy data on the vendor's computers. This would be somewhat self-defeating, as it could easily leave the vendor - especially a small vendor - in a position where they are unable to fix any problems and satisfy their consumers. In addition, destruction is a one-time affair; once an affected computer is restored, the vendor's incentive to fix problems dissipates. Such an I4NI system, as a contractual agreement, would likely be unenforceable anyway, because data destruction even under these circumstances might be deemed illegal.

An alternative would be an I4NI system with a lingering effect, an effect which would be annoying to the vendor but not devastating. For example, an activated I4NI system could slow down a company's Internet access, which might hamper online sales and communication, but allow problems to be fixed by the vendor internally. This kind of system could actually create a new market opportunity: companies which specialize in I4NI systems, or *I4NI providers*. An I4NI provider would supply infrastructure for conducting voting, and co-locate some hardware at a vendor's Internet access point; they could use this hardware to enforce an activation penalty.

---

[2]Read: "Eye for an Eye."

Both of the above I4NI systems are punitive in nature. A third possibility is analogous to real-world elections. A vendor is given an operating "lease," which is renewed at periodic intervals by satisfied consumers (or not renewed by dissatisfied consumers, as the case may be).

Regardless of the implementation, the presence of an I4NI system would give a vendor very good reason to be attentive to their consumers' satisfaction. Generally, I4NI systems could be used to provide other incentives to the vendor besides software quality, such as improved consumer service or better technical support.

Discussion

Surprisingly, while there are no I4NI systems currently in existence, there is work related to I4NI systems. Periodically, some vendors will have contests, offering prizes to people who can breach the security of a particular piece of software (Ilett, 2004). Presumably a vendor offering a contest observes what attackers do, and uses that information to improve their software. Here, as in I4NI systems, the vendor is a willing participant in the process, but the contests are for a limited time and do not provide an ongoing incentive to the vendor.

Another related mechanism is the "Usenet Death Penalty," or UDP (Lucke, 2003). A UDP can be imposed by consensus onto an uncooperative Usenet site: one posting spam to newsgroups, for instance. A targeted site will have its Usenet traffic dropped, effectively cutting it off from the rest of the world, until such time as the problem is fixed. Targeted sites are given five business days' notice of an impending UDP so that it may be forestalled if possible. Real-time blackhole lists (RBLs) for refusing mail from spamming or spam-friendly sites are related technology, too, but RBLs have come under fire for perceived biases and the opacity of their decision-making processes (Jacob, 2003). The target is not a willing participant in the case of both UDPs and RBLs, unlike I4NI systems.

I4NI systems can also be analyzed from technical, ethical, and economic standpoints.

Technical

One of the biggest issues in deploying malware for any legitimate purpose is containment. How can we ensure that the malware used in an I4NI system does not escape the vendor's machines? We suggest three measures:

1. Only non-propagating malware is used, meaning that the malware cannot leave the vendor's machines of its own accord, and any escape must be due to deliberate copying. In other words, I4NI malware has the same chance of escaping on its own as Microsoft Word does.

2. The malware must be installed on the vendor's machines under strict copyright terms; deliberate copying would be considered software piracy and infringement upon the copyright.

3. The I4NI malware should be constructed so that it *only* works on the vendor's machines. As suggested by Filiol (2004), the malware can be encrypted using strong encryption algorithms and a key unique to the vendor; the key could be the vendor's IP address, for example.[3] The main body of

[3]The IP address is not necessarily unique, of course, but it is used here for illustration.

the I4NI malware could then only be successfully decrypted and run if the malware was running on a machine with that same IP address. Other safety checks could be embedded into the I4NI code too.

Nuclear disarmament and I4NI systems share a common problem: verification. A vendor who has agreed to an I4NI system must not be able to avoid its activation. There must be some way to ascertain that I4NI malware is installed and ready, and - should activation be required - functioning. Having separate, co-located hardware supplied by an I4NI provider would be ideal, as suggested in the last section. Any software solution, however, must run in the hostile environment of the vendor's computers. The use of strong encryption mentioned above could be used to hide the specifics of an I4NI system until it is activated; in general, any anti-anti-virus techniques could be used, especially anti-emulation techniques. For example, it might be useful to know whether or not the I4NI malware was running inside a virtual machine like VMware (z0mbie, 2002).

A danger with I4NI systems is that the vendor could be incorrectly blamed for an unrelated problem with consumers' computers. Computer systems are complex enough that it can be difficult to precisely pinpoint the cause of a system problem. Here, the scale of I4NI is important. Given enough consumers with slightly different system configurations, a persistent vendor software problem should stand out. This analysis could simply involve consumers talking in an open (electronic) forum, or an automated system could be constructed, which would take all consumer system configurations and try to isolate a common problem.

Another scenario to consider is a vendor who has multiple I4NI systems installed, and whether or not the different I4NI systems could affect one another. This may be a situation best dealt with centrally by an I4NI provider.

Finally, what is the role of anti-virus software in an I4NI system? Probably none: I4NI malware is legitimate software which the vendor already knows is present. At best, it would fall into the realm of gray area detection, perhaps warranting a warning.

Ethical

At first glance, it might seem tempting to assess I4NI systems from a deontological point of view.[4] In other words, we could argue that "malware must never be used" is a universal rule, and consequently I4NI systems which use malware are unethical. However, this ham-handed approach treats all malware the same, regardless of intent, implementation, or effect.

In fact, I4NI malware is a new class of malware. It is not malware in the traditional sense, because the vendor has agreed to its installation and possible activation. It is also not benevolent malware in the way that Cohen (1991) intends, as I4NI malware clearly has a malicious effect if activated.

Concocting a rule which captures these nuances of I4NI malware would result in a rather ungainly construction, which would not be easy to reason with. Rule-utilitarianism would suffer from a similar problem. For this reason, we analyze I4NI systems using act-utilitarianism, where an act is ethical if it produces the most "utility," or good consequences, for everyone involved (Bentham, 1789).

---

[4]Defining the horrendously-long terms used in ethics is beyond the scope of this paper, but there are a number of excellent references on technology and ethics, such as Baase (2003) and Martin and Schinzinger (1996).

There are really two separate acts to consider. First, is it ethical to install I4NI systems at all? The vendor is a willing participant, so this is more a consideration of whether or not installing an I4NI system can affect nonparticipants. As described in the last section, the malware cannot escape the vendor's machines on its own, and even if it should get out, it will not be able to run. Therefore, no bad consequences happen to nonparticipants, and no unexpected consequences happen to participants, so we must conclude that installing I4NI systems is ethical.

Second, is it ethical to activate I4NI malware? We need to look at the utility for all parties involved. (We assume that activation is designed so that the vendor can still work on fixing the software's problems.)

1. The vendor. Despite short-term ill effects, activation will provide strong incentives for the vendor to improve their software and, consequently, their position in the market. The vendor thus has positive utility in the long term.

2. Consumers who voted for activation. This group of people is getting the activation they voted for in the short term, and better software in the long term. They have positive utility throughout.

3. Consumers who voted against activation, or who didn't vote. In the long term, these consumers will receive the same benefits as those who voted for activation. Even if this group were to have negative utility in the short term, there must be more people with positive utility in the short term: the consumers who voted for activation. The majority voting thus ensures that there is more positive utility then negative overall in the short term.

4. Investors or stockholders in the vendor, if any. People with a vested interest in the vendor's success will benefit in the same way that the vendor does with improved software, giving long-term positive utility.

5. Others. The non-propagating I4NI malware could only affect others as a result of the action that occurs when it is activated. A DDoS attack on the vendor, for instance, would not be acceptable because it impacts network traffic for others; the same argument applies to attempts to flood the vendor's machines with traffic (short of a full DDoS) like Lycos did recently (Libbenga, 2004). So long as the I4NI malware's action is appropriately chosen, other parties are not affected and do not change the overall utility.

Overall, the utility on both the short and long term is positive for all parties involved, so activation of I4NI systems is ethical as well. However, being ethical and being economically viable are two different things.

Economics

Even though quality plays a large part in how successful a product is in a mature market (Card, 1995), it is not clear how consumers can judge attributes like software quality and security, especially prior to purchasing it (Gehring, 2002). Consumers essentially have to assess quality and security using the *absence* of indicators: quality is when the software doesn't crash; security is when the software isn't compromised. Compared to a new user interface or some added feature, software quality and security is invisible, and as such is not a direct market influence.

Herein lies the problem. The vendor wants to gain a competitive advantage in a mature market, and differentiate their product from others, by demonstrating that their software is of high quality. The consumer wants to ensure that they get high-quality software, and that the vendor will be responsive to any problems.

Using an I4NI system fulfills the needs of both vendor and consumer. The vendor, by using such a system, is making a public statement: we are confident that our software is high-quality. The consumer is empowered to respond in a direct, democratic way if that public statement is false, and give the vendor an incentive to fix the problem. There is a danger, though, that I4NI systems may act as a disincentive to dissatisfied consumers who might otherwise choose another software vendor.

Cost and price

The implementation of the I4NI mechanism requires software development, monitoring, and ancillary services. This will result in increased production costs, which in turn will be added to the consumer's price. The price increment impacts negatively on the success of the product, and can only be sustained under certain conditions.

1. Consumers must be willing to pay more for a higher quality product. Consumers are probably willing to accept increased prices in exchange for higher quality: psychological mechanisms and atypical demand curves are known to exist, particularly for high level goods or services (Monroe, 1973). Additionally, the empowerment granted to the consumer, emphasized by advertisement, may have a positive psychological effect, giving individual consumers the illusion of having power over the vendor.

2. The price can increase only up to a threshold. The threshold is the price at which another vendor can offer an equivalent or better product, which would defeat the competitive advantage of the I4NI vendor.

3. Production costs in excess of the threshold must be absorbed by the vendor. If the incremental cost of producing an I4NI system is higher than the production cost of a product of equivalent quality, the mechanism is not economically valuable.

Consumer groups

A vendor's customers can be viewed as an interest group; indeed, this group may be an entity in its own right, characterized by a vested interest and a substantial power, ultimately capable of affecting a market. These individuals can be compared to shareholders in any other business, where each buyer has an interest in the long term quality of the vendor's software. The amount of an individual's interest in the software quality is equivalent to the ownership of a number of shares of a business: one share per software license purchased.

The consumer group can make decisions, vote, and form coalitions like any other shareholder group. They can set their own rules, governing aspects such as:

- Frequency of meetings (physical or virtual)

- Voting mechanisms (representation, quorum, majority)

- Membership (entrance, exit, loss of right in the case of prolonged absence from meetings or discontinued use of the software)

Alternatively, the rules can be set by the I4NI provider. While limiting the autonomy of the consumer group, this alternative may include mechanisms to protect the vendor. For example, one large consumer (a company or corporation) may own enough licenses to able to affect, by itself, the decision of the entire group; an I4NI provider could prevent the vendor's competitors from playing this game.

Market: the demand side

In a typical good or service market, the number of consumers is so large that no individual can affect the price of a product. Conceptually, this corresponds to perfect competition on the supply side of the market: the number of suppliers is so high, and the individual's contribution so minor, that no individual can affect the price (Eaton et al., 1999).

A consumer group empowered by the I4NI mechanism has the potential of acting on the market as what could be called a *demand side oligopoly*, which may take the form of a *demand side cartel*. The feasibility of this anomalous entity depends inextricably on the characteristics of the product and both sides of the market.

On the supply side, the consumers' counterpart should be a monopoly or a very collusive oligopoly. This is necessary for two reasons: if there are several independent suppliers the consumers will be split among them, and in a competitive supply market the power of the consumer group may be tempered by competitive dynamics.

On the demand side, consumers should be relatively few, have a strong interest in the product, and possess the will and ability to communicate with one another in order to implement a strategy.

These supply and demand conditions are strongly connected with the nature of the product. Consider the following examples.

1. The word-processing market. Supply side: one supplier virtually preempts the market, forming an almost a perfect monopoly. The condition is met. Demand side: the consumers are such a vast and heterogeneous set of individuals that any form of collusion is hardly imaginable. Again, the condition is met.

2. The market of more specialized packages, e.g., CAD systems. Supply side: relatively few, but independent and competitive vendors. The condition is not met. Demand side: the consumers are relatively few, homogeneous, and capable of communicating - the condition can be met.

3. The market of highly specialized software, e.g., a package for petroleum exploration, or for a sophisticated medical procedure. Supply side: monopoly or collusive oligopoly. The condition can be met. Demand side: small number, homogeneity, and mutual acquaintance of consumers - once again, the condition can be met.

The conditions for collusive behavior on the consumers' part are restrictive, and even in example 3 the consumers must be willing to collaborate, rather than trying to work out individual agreements with the vendor.

Market: the supply side

The creation of new jobs and revenues resulting from I4NI development, and the establishment of I4NI providers, represents a potential spin-off effect. As we noted earlier, software defects generate revenue through bug fixing (Barnes, 2004). If the problems are fewer and solved in a more efficient way, the flow of revenue and jobs may be lost or reduced. The net effect on the job market should still be positive, with long term consequences at the macroeconomic level, particularly on salaries and demand.

The rise of a new sub-industry in the software market will result in a redistribution of revenue and market share, whereby established software vendors are forced to renounce a portion of their power and employees in favor of the I4NI providers. A greater fragmentation in the software supply market will increase the competition on that market. Whether a more competitive market is desirable can be debated, but antitrust laws in the US and Europe may be viewed as a positive indication.

I4NI Variations

There are a few interesting variations and applications of I4NI systems.

A vendor can temporarily, rather than permanently, fix a problem. A software problem might be too difficult to repair in a current version, but is fixed in a not-quite-ready software release. A vendor might somehow bribe consumers to buy time until the next release, perhaps with rebates, merchandise, or small shiny objects. In the end, the consumer is still satisfied, so this type of short-term vendor workaround would proffer the same market advantages.

Other uses might be found for the consumer empowerment resulting from an I4NI system. For example, the release of technical documentation might be coerced from an obstinate vendor, or the vendor might be forced to alter the price of the next software release. Such applications could easily be construed as extortion, though, so valid reasons to activate an I4NI system must be clearly set in advance.

Finally, a competitor could conceivably buy their way into a voting majority position by purchasing many licenses of a vendor's product, then activating the vendor's I4NI system to gain an unfair advantage. Such anticompetitive acts could be countered legally.

Other Markets for Malware

There are other possibilities for malware-related markets which are either legitimate or quasi-legitimate. This qualification excludes markets which involve illegal activities; for example, there are strong indications of cooperation between spammers and malware authors. Malware is used to create a "botnet" of zombie machines, which can then be used to send spam (LURHQ, 2003), and botnets-for-hire are available to spammers for a fee (Acohido & Swartz, 2004). However, establishing a botnet involves compromising machines - an illegal activity now in many countries - so we do not consider this market further.

Selling malware, particularly viruses, has been suggested in jest (Marlatt, 2001) but has actually occurred. Virus libraries/collections of varying degrees of quality have been sold in the past (Bontchev, 1993; Tarala, 2002), and it is still possible to find collections offered for sale (American Eagle, 2004).

Slightly further afield, Kannan and Telang (2004) explore the economics of establishing markets for vulnerability disclosure. Malware, though, can be seen as an application of software vulnerabilities at best, and vulnerability information is currently freely-available on full-disclosure mailing lists and other venues.

Malware skills, if not malware itself, are also in legitimate demand. It is no longer in the realm of conspiracy theory to suggest that military and government organizations are developing offensive information warfare capabilities involving malware. China, Taiwan, North Korea, and Singapore are known or suspected to have this capability (Bristow, 2000), as are Cuba and Bulgaria (Messmer, 1999) and Russia (Thomas, 1996). The United States has openly announced recently that it will be developing information warfare attack capabilities too (Caterinicchia, 2003; Tiboni, 2004).[5] It is not clear if military and government organizations would overtly employ malware authors, but the same cannot be said of the private sector: Sven Jaschan, the alleged Sasser worm author, was offered a job by Securepoint (Best, 2004).

Conclusion

Malicious software, in the form of I4NI systems, can give strong incentives to software vendors to improve the quality of their software. I4NI systems are both ethical and safe, and empowering consumers in this manner has interesting economic ramifications. Vendors benefit by using I4NI to separate their product from competing ones in a mature market. Who will be the first to sign up?

Acknowledgments

The authors' research is supported in part by grants from the Natural Sciences and Engineering Research Council of Canada. Shannon Jaeger made a number of helpful comments on this paper, as did the anonymous referees.

[5]To be fair, the United States has not specifically announced that it will use malware, but malware is well within the realm of possibility for electronic warfare (Cramer & Pratt, 1990).

References

Acohido, B., & Swartz, J. (2004, 8 September 2004). Going price for network of zombie PCs: $2,000-$3,000. USA Today, p. B04.

American Eagle Publications. (2004). Outlaws of the Wild West Computer Virus CD-ROM. Retrieved 16 December, 2004, from http://ameaglepubs.com/store/outlaws.html

Baase, S. (2003). A Gift of Fire, 2nd edition. Prentice Hall.

Barnes, D. A. (2004). Deworming the Internet. Texas Law Review 83(1), pp. 279-329.

Bentham, J. (1789). An Introduction to the Principles of Morals and Legislation. (Excerpted and reprinted in T. K. Hearn, Jr. (Ed.), Studies in Utilitarianism, 1971, Meredith Corporation, pp. 15-38.)

Best, J. (2004, 20 September 2004). Security firm looks to hire alleged Sasser author. CNET. Retrieved 17 December, 2004, from http://news.com.com/2102-7349_3-5374636.html?tag=st.util.print

Bontchev, V. (1993). Analysis and Maintenance of a Clean Virus Library. Proceedings of the 3rd International Virus Bulletin Conference, pp. 77-89.

Bristow, D. (2000, 1 December 2000). Asia: grasping information warfare? Jane's Intelligence Review. Retrieved 17 December, 2004.

Card, D. N. (1995). The RAD Fad: Is Timing Really Everything? IEEE Software 12(5), pp. 19-22.

Caterinicchia, D. (2003, 7 February 2003). DOD plans network attack task force. Federal Computer Week. Retrieved 17 December, 2004, from http://www.fcw.com/fcw/articles/2003/0203/web-net-02-07-03.asp

Cohen, F. (1987). Computer Viruses: Theory and Experiments. Computers & Security 6, pp. 22-35.

Cohen, F. (1991). A Case for Benevolent Viruses. Retrieved 22 December, 2004 from http://all.net/books/integ/goodvcase.html

Cowan, C., & Pu, C. (1998). Death, Taxes, and Imperfect Software: Surviving the Inevitable. Proceedings of the 1998 Workshop on New Security Paradigms, pp. 54-70.

Cramer, M. L., & Pratt, S. R. (1990). Computer Viruses in Electronic Warfare. Retrieved 17 December, 2004, from http://iw.windermeregroup.com/Papers/virus_ew.html

DePompa, B. (2004). "DETER" fills IT security testing void. Government Security News. Retrieved 13 December, 2004, from http://www.gsnmagazine.com/nov_04/deter_program.html

Eaton, B., Eaton, D., & Allen, D. (1999). Microeconomics. Scarborough: Prentice Hall Canada.

Filiol, E. (2004). Strong Cryptography Armoured Computer Viruses Forbidding Code Analysis: the BRADLEY virus. Research report 5250, INRIA, June 2004, 10pp.

Fisk, M. (2002). Causes & Remedies for Social Acceptance of Network Insecurity. Workshop on Economics and Information Security, 4pp.

Fraud and Related Activity in Connection with Computers, 18 U.S.C. 1030 (1996).

Free Software Foundation. (1991). GNU General Public License, version 2, June 1991. Retrieved 17 December, 2004, from http://www.gnu.org/copyleft/gpl.html

Gehring, R. A. (2002). Software Development, Intellectual Property Rights, and IT Security. First Workshop on Economics and Information Security.

Ilett, D. (2004, 27 September 2004). E-mail firm baits hackers with security challenge. ZDNet. Retrieved 20 December 2004, from http://news.zdnet.com/2100-1009_22-5383988.html

Jacob, P. (2003, 3 January 2003). The Spam Problem: Moving Beyond RBLs. Retrieved 20 December 2004 from http://theory.whirlycott.com/~phil/antispam/rbl-bad/rbl-bad.html

Kannan, K., & Telang, R. (2004). An Economic Analysis of Market for Software Vulnerabilities. The Third Annual Workshop on Economics and Information Security (WEIS), 12pp.

Libbenga, J. (2004, 26 November 2004). Lycos screensaver to blitz spam servers. The Register. Retrieved 26 November 2004 from http://www.theregister.co.uk/2004/11/26/lycos_europe_spam_blitz

Lohr, S. (2003, 6 October 2003). Product Liability Lawsuits Are New Threat to Microsoft. New York Times (Late Edition), p. C2.

Lucke, K. (2003, 16 November 2003). Usenet Death Penalty FAQ v1.1.1. Retrieved 18 December 2004 from http://www.stopspam.org/faqs/udp.html

LURHQ Threat Intelligence Group. (2003, 21 April 2003). Sobig.a and the Spam You Received Today. Retrieved 16 December, 2004, from http://www.lurhq.com/sobig.html

Marlatt, A. (2001) ViruSystems: Can Anyone Make Money Making the Bug? SatireWire. Retrieved 13 January, 2005, from http://www.satirewire.com/ebow/virusystems.shtml

Martin, M. W., & Schinzinger, R. (1996). Ethics in Engineering, 3rd edition. McGraw-Hill.

Messmer, E. (1999). Threat of 'infowar' brings CIA warnings. Network World Fusion. Retrieved 17 December, 2004, from http://www.nwfusion.com/archive/1999/75306_09-13-1999.html

Monroe, K. B. (1973). Buyers' subjective perceptions of price, Journal of Marketing Research, 10 (1), pp. 70-80.

Pfleeger, C. (2004, 26 November 2004). You Get What You Pay For: Why We Have So Many Security Problems with Software. Talk given at the University of Calgary, Department of Computer Science.

Schneider, V. (1988). Approximations for the Halstead software science software error rate and project effort estimators. ACM SIGPLAN Notices 23(1), pp. 40-47.

Schneier, B. (2004). Information security: How liable should vendors be? ComputerWorld. Retrieved 15 November, 2004, from http://www.computerworld.com/printthis/2004/0,4814,96948,00.htm

Schneier, B. (2001). Insurance and the Computer Industry. Communications of the ACM 44(3), pp. 114-115.

Shoch, J. F., & Hupp, J. A. (1982). The "worm" programs - early experience with a distributed computation. Communications of the ACM 25(3), pp. 172-180.

Tarala, J. (2002). Virii Generators: Understanding the Threat. SANS InfoSec Reading Room. Retrieved 16 December, 2004, from http://www.sans.org/rr/whitepapers/malicious/144.php

Thomas, T. L. (1996). Russian Views on Information-Based Warfare. Airpower Journal, Special Edition, pp. 25-35.

Tiboni, F. (2004, 14 December 2004). Air Force seeks cyberwar edge. Federal Computer Week. Retrieved 17 December, 2004, from http://www.fcw.com/fcw/articles/2004/1213/web-cyberwar-12-14-04.asp

Wang, H., & Wang, C. (2003). Taxonomy of Security Considerations and Software Quality. Communications of the ACM 46(6), pp. 75-78.

z0mbie. (2002, 6 July 2002). VMware has you. Retrieved 20 December 2004 from http://z0mbie.host.sk/vmware.txt