"Software Patents" - IT-Security at Stake?

Robert A. Gehring, TU Berlin (rag@cs.tu-berlin.de)

Short abstract

This paper in short[1] presents the thesis that insecurity of software is due to interaction of technological and legal shortcomings, fostered by economic rationality. Ineffective liability laws further the distribution of unreliable and insecure software. Copyright protection for software hinders the quality improvement. Patent protection encourages the use of proprietary instead of standard technology. Open source development is proposed as a starting point for a risk management strategy to improve the situation. Existing patent laws need therefore be modified to include a "*source code privilege*".

Introduction

New legislative measures are regularly complemented by technical measures developed by hardware and software producers to enhance the safety of the intellectual property of their respective owners as well as to enable new business models such as pair-per-use. To the technical measures themselves legal protection is given by corresponding accommodated laws, often brought about by private law making procedures: *«[T]he tradition in copyright legislation involves getting a bunch of copyright lawyers to sit at a bargaining table and talk with one another [...]»* (Litman 2001: 31)

The problem of how the security of information technology in general is affected by laws and technical measures is somewhat out of focus, at least when considering the political and legal process that lead to the new legislation. Lobbyists of the right holders put heavy pressure on the politicians to enact laws

---

[1] Short version as contributed to the international conference "Innovations for an e-Society. Challenges for Technology Assessment", held at Berlin, Germany, October 17-19, 2001. A full length version is available on the internet via http://ig.cs.tu-berlin.de/.

to protect their commercial interests. As the topic of security is approached, the right holders worry about the safety of their respective intellectual property assets but actually don't care much about the security of the underlying information infrastructure.

Technical reasons for insecure software

A software product development starts with a more or less detailed description of the tasks the program has to do. Such a description is derived from a systems analysis of the environment where the software is to be deployed. The functional specification presents the building plan for the software.

In addition the input/output-relation of every functional part of the program should be described in order to derive sufficient test instructions thereof. Measured by real world conditions, a functional specification is sometimes incomplete with respect to functional requirements. And it is nearly always incomplete when it comes to security considerations: «[M]odern systems have so many components and connections—some of them not even known by the systems' designers, implementers, or users—that insecurities always remain.» (Schneier 2000: xii) No serious programmer would claim that software is secure because of it is tested to be correct:

«The developers are so in tune with what it *should* do, they cannot see what it *might be able* to do.» (Pipkin 2000: 75)

Besides the unavoidable blind spots in the functional specification, there is another problem in the process of writing software: software is designed and written by humans. And as with every writing human beings are involved in, mistakes are made and need to be found and fixed. It is a limitation in the development process owed to the imperfect human mind.

Not all errors can be found through testing. Thus—in a faulty reaction—the testing procedure can only show the presence of errors in the program. But it

cannot show the absence of errors.(Floyd 1997: 664; Kaner 1997) This is an unavoidable technical limitation of the testing process.

Economic reasons for insecure software

Above all, proprietary software is a product for a market. Cost-benefit analysis are an integral part of the development process. If the expected costs of liability in sum are lower than the expected costs of a more complete testing and debugging process, to deliver an unsafe product will be preferred by the software producer.

Network externalities have the greatest influence on the behavior of software producers.(Shapiro/Varian 1999) A software distributor has to bear in mind that software as a good applied in virtual networking environments is affected by positive feedback effects: The more customers use the same software product the greater is the value of the software to the individual user.

Positive feedback works to the advantage of the largest supplier. Regarding the market, positive feedback cycles often end in a "winner-take-all" situation. (Shapiro/Varian 1999) Incentives to be the first on the market and to establish one’s own products as de facto standards are very high. Faster and less thorough testing procedures allow for a shorter time-to-market, thus leading to a competitive advantage. In network economics, controlling standards is of greater importance to a commercial success than to deliver a better product.

Asymmetric information between producers and buyers within the software market makes another contribution to the problem of flawed and insecure software. It cannot be the business interest of a software producer to provide information about weaknesses of his product to potential customers before they buy it.

Because reverse engineering is declared an unlawful activity by copyright laws, there cannot be a provable serious source of quality information to serve as a basis for rational consumer choice. A competition for quality is disabled as long as reverse engineering software distributed in binary form is deemed unlawful.

Certification of products, often proposed as a solution to this dilemma, won't work if neither the producer of the software nor the certification authority will have to bear the costs of ill-certified software. Instead, as security expert Ross Anderson explained (Anderson 2001), the certification process probably will be—and in reality quite often is—adapted to the needs of the software producer.

That is the simple economic rationale behind commercial software development and its built-in preference for insecure software: It is a perfectly rational behavior for a commercial software producer to distribute unsafe products as long as it is to his advantage.

Legal obstacles to better software quality and security

From a legal point of view, software is treated as some kind of literary work. (Raskind 1998) There is no special liability law to be applied in cases involving mass market software.

Instead of the development of a sui generis law for software that would have equilibrated the interests of software producers, software users and of the public of course, existing laws have been extended in order to cover the demands of software producers solely.

Copyright law protection for software contains a broad ban—with only a few exeptions—on reverse engineering. It is unlawful to reconstruct a human readable form from the binary code. Reverse engineering is made illegal even for most honest purposes.[2] There is no exception for security inspection and/or enhancement.

The prohibition of reverse engineering furthers the above mentioned market intransparency. Thus it hinders the development of a market for software security.

---

[2] For the consequences see, e.g. (Kaner 1998).

The second important legal hindrance to the enhancement of quality and security of software products consists in the increasing patent protection for software.

Software can in part be protected by patenting its technology. Patent laws give the patent holders the exclusive rights to the patented technology—in every possible implementation—and do not allow the offering of compatible technology without a license. Patent protection often bars competitors from the market if core parts of a standard technology are protected by patents. Since network externalities have great influence, the incentives are high, not to license technology to competitors.

Patent protection for software has implications for IT security.

Because of the absolute legal protection that patent law provides, compatible technology from a competitor may be blocked. A faulty implementation of a certain technology may well become the single one solution available on the market.

Secure technology itself may be patented. In such cases, no software producer is allowed to include functional equivalent technology within his products without license. Thereby the fast spreading of secure technology can be hindered.

Business models with a core idea of securing systems may be patented. Actually, it already happened.[3] In effect, one has to acquire a license in order to make systems safe or to fix security flaws in a certain way – regardless of a possible emergency. Rarely applied, compulsory licensing rules provided hitherto no solution.

---

[3] See, e.g. Finjan Software, Inc. (San Jose, CA), U.S. Pat. 6,167,520 (26 Dec 2000): System and method for protecting a client during runtime from hostile downloadables; McAfee.com Corporation (Santa Clara, CA), U.S. Pat. 6,266,774 (24 Jul 2001): Method and system for securing, managing or optimizing a personal computer.

We can conlude that the more software technology is protected by patents, the higher is the probability for certain faulty software products to become very common. Until today, there is no legislative answer to the mentioned problems.

Risk management

In view of the error-prone development process which leads to faulty software, which in turn leads to insecure systems, it is time to ask for an adequate *risk management strategy* to cover the public interest in system security.

Such a *risk management strategy* has to be constructed in a manner that it will reduce the risks coupled with the use of software in the long term. That means to reduce the number of errors in the code, to reduce the scale of security weaknesses and to minimize the harmful consequences of security breaches.

The best method we know so far to enhance the quality of software is the deployment of well tested standard components combined with a process of peer review by experts. The creation of more secure software requires incremental improvement in order to fix detected weaknesses. (Sommerville 2001: 566) Peer review plays a crucial role in this *quality management process*.

The delay between the detection of an error or security weakness and the availability of a service pack depends solely on the suppliers subjective estimation of its relevance. From a security point of view, the delay should be kept as short as possible. Even better if the user could do the repair on his own.

Security needs to be thought of as a «process». *«And if we're ever going to make our digital systems secure, we're going to have to start building processes.»* (Schneier 2000: xii)[4] And that process-building must be kept alive over the time a software product stays in use. The security process must again and again be adapted to reflect changing environmental conditions and experiences. Environmental conditions are nowhere the same. We have to realize that there

[4] Similarly (Pipkin 2000, xx).

simply cannot be a one-fits-all solution. Security is to be tailor-made to reach the required level of efficiency.

To the current knowledge, there is only one development process that can fulfill the mentioned quality requirements and at the same time supply the basis of a security process within a risk management strategy. This is the open source software development model.[5]

Since the source code of the programs is publicly available, there is no need for reverse engineering. To be able to understand how the program works one can simply read the source. This alone is not the solution to reliability and security problems. Rather it is the decisive technologic prerequisite a number of security experts demands in order to make secure systems available. (Schneier 2000: 343f; Pfitzmann et al. 2000)

Availability of source code enables users at home and at work to fix security weaknesses as soon as they become aware of it.

Open source software spurs competition. The use of open standards removes the bias in favor of a dominant market player with proprietary technology. Other competitors with more reliable products get a chance and the unwelcome results of network externalities can be minimized. And the market transparency grows. A producer-independent certification process could be established and meet the users' needs.

Last but not least, a huge amount of open source, well scrutinized code is available without royalty fees. Secure software and services can be developed out of it at low costs.

Given all these prospects, open source software shows reasonable qualities to be preferred as a development and distribution model in comparison to proprietory

---

[5] I will use the term open source in a generic manner and not differentiate between *free software* and *open source software*. The interest is directed to the technical and legal possibilities and from this point of view both, free and open source software provide similar features.

ones. But this model is in danger. The goldrush in software patenting may well stall what looks so promising. There are serious problems connected to the patent protection for software.

Patent protection prevents the use of patented technology within open source software without the appropriate license. Put simply, patent law favors those who try to hide patent infringing code through binary distribution.

The average open source programmer without support from a patent department as large software producers have it at their hands, is not in the position to avoid writing code that possibly may infringe someone else's patent claims. He is financially not in the position to defend against patent litigation, even if the complaints are unreasonable.

To summarize: The open source software development process encourages the use of the best software engineering principles we know today. High quality and security of software and computer systems can thereby be achieved. Open Source software encourages the competition for better security. However, software patents present a real threat for the open source software development and distribution model. Unrestricted possibilities of enforcing patent against open source devolopers could mean to put IT security at stake.

Perhaps, the proposal of a "**source code privilege**", may present a way to a solution. (Lutterbeck et al. 2000)

The core proposal suggests:

> «The use of the source codes of computer programs must be granted privileged status under patent law. The creation, offering, marketing, possession, or introduction of the source code of a computer program in its various forms must be exempted from patent protection (source code privilege).» (Recommendation PP-1).

References

Anderson, Ross: **Why Information Security is Hard – An Economic Perspective**, 2001, on the internet: http://www.cl.cam.ac.uk/ftp/users/rja14/econ.pdf [28 Aug 2001].

Floyd, Christiane: **Softwaretechnik** [Software engineering], 14.2.1 Eigenschaften von Software [Properties of software], in P. Rechenberg, G. Pomberger: Informatik–Handbuch, Carl Hanser Verlag, München, Wien, 1997.

Kaner, Cem: **The Impossibility of Complete Testing**, in SOFTWARE QA, Volume 4, #4, p. 28, 1997, on the internet: http://www.kaner.com/articles.html [28 Aug 2001].

Kaner, Cem: **The Problem of Reverse Engineering**, in SOFTWARE QA, Vol. 5, #5, 1998, on the internet: http://www.kaner.com/articles.html [28 Aug 2001].

Litman, Jessica: **Digital Copyright**, Prometheus Books, Amherst, NY, 2001.

Lutterbeck, Bernd; Horns, Axel H.; Gehring, Robert A.: **Sicherheit in der Informationstechnologie und Patentschutz für Softwareprodukte - ein Widerspruch ?** (Security in Information Technology and Patent Protection for Software Products: A Contradiction?, Short Expertise Commissioned by the Federal Ministry of Economics and Technology), Berlin, Dec 2000, on the internet: http://www.sicherheit-im-internet.de/download/Kurzgutachten-Software-patente.pdf [28 Aug 2001].

Pfitzmann, Andreas; Köhntopp, Kristian; Köhntopp, Marit: **Sicherheit durch Open Source? Chancen und Grenzen**, in Datenschutz und Datensicherheit 9/2000, pp 508-513.

Pipkin, Donald L.: **Information Security**, Prentice Hall PTR, Upper Saddle River, NJ, 2000.

Raskind, Leo J.: **Copyright**, in The New Palgrave Dictionary of Economics and The Law, Vol. 3, p. 478ff, Macmillan Reference Ltd., London, 1998.

Schneier, Bruce: **Secrets and Lies. Digital security in a networked world**, John Wiley & Sons, Inc., New York, 2000.

Shapiro, Carl; Varian, Hal R.: **Information rules. A strategic guide to the network economy**, Harvard Business School Press, Boston, MA, 1999.

Sommerville, Ian: **Software Engineering**, 6th edition (german translation), Pearson Studium, 2001.