Security: Source Access and the Software Ecosystem

Craig Mundie, CTO, Advanced Strategies and Policy
Microsoft Corporation

In recent years, we have seen computing play an ever-increasing role in almost every aspect of society, enriching individual lives and transforming the way companies of all sizes do business. Within the next decade Internet access will become nearly universal in developed nations.[1] As society taps the vast potential of computing by ever more pervasive means, the issue of security carries greater importance.

Today, the technology industry, governments and researchers are engaged in a significant debate over the best ways to ensure that tomorrow's computing systems are unfailingly reliable and secure. This debate is lively and intense, and rightly so – if we cannot ensure the highest possible level of security, we will not make the most of computing technology's promise.

Within the software industry, there is a concurrent debate regarding the implications of open source software, free software and commercial software. The discourse surrounding these differing approaches covers a broad spectrum of issues such as the intellectual property rights associated with software, open industry standards, business and licensing models and the security of software produced and distributed under these models.

The goal of this paper is to explore the relationship between the security of software and the model under which that software was produced and distributed. Additionally, this document will outline Microsoft's position that ensuring security is a multi-dimensional process. Within the context of the development model employed in the creation of software, broad source code access ultimately has a neutral effect on the security capabilities of that software.

The Software Ecosystem

The tremendous progress computing has made in the past fifty years is the product of a vast "ecosystem" of commercial software companies, government and academic researchers[2] each playing a unique and critical role in advancing the state of the art.

---

[1] Business Software Alliance, "Confident of Technology's Future: Executives Look Ahead," December 2001, 4-5, at www.bsa.org/resources/2001-12-03.72.pdf (visited Feb. 6, 2002).

[2] For purposes of simplification in conveying the cyclical nature of relationships within the software ecosystem, the roles of government and academia here are somewhat understated. In fact, U.S. federal agencies are required by law to encourage certain grant recipients and public contractors to patent the results of government-sponsored research, and universities are quite active in asserting research-related IP rights. Since at least 1980, the U.S. government has pursued a vigorous policy of transferring the results of federally funded technology research to industry to promote innovation and commercialization. *See, e.g.,* the Bayh-Dole Act of 1980, the Stevenson-Wydler Technology Innovation Act of 1980, the Federal Technology Transfer Act of 1986, Executive Order 12591 of 1987, the National Technology Transfer and Advancement Act of 1995, and the Technology Transfer Commercialization Act of 2000.

Working with a wide range of motives, development models and approaches to ownership and accountability, the ecosystem collectively produces innovative new technologies, creates jobs and business opportunities, and has increasingly become a major contributor to the global economy.

The result of this software ecosystem is a cycle of sustained innovation. Governments and universities undertake basic research and share this knowledge with the public; companies in the private sector build on this base of knowledge to create commercial products, while also contributing to the work of common standards bodies and undertaking further research. Their success leads to greater employment, tax revenues and additional funding of academic research projects. Throughout the life cycle of a technology from research through commercialization, many different software development models may be employed.

The open source software (OSS) development model has been widely used in government and academic research over the past 20 years.[3] Many technologies that originated in OSS development have been used in conjunction with software produced by the commercial software industry, and the result has been the explosive advances in the capability, usability, accessibility and affordability of software.

The successful development of a new technology often incorporates a number of different development models and commercial approaches – TCP/IP, for example, started as a government research project, matured in academia under the OSS development model and evolved into an open industry standard. It was then further refined and brought into the mainstream of computing through proprietary implementations by commercial software companies such as Novell, Apple, IBM and Microsoft. Microsoft's Windows operating system, on the other hand, was developed privately and for profit, but implements dozens of open industry standards and published application programming interfaces that are the key not only to its success, but have created opportunities for more than one hundred thousand companies to make money writing software that interoperates with the Windows platform.

An Evolving Security Environment

As the computing environment has evolved from stand-alone mainframes and desktop PCs to a vast, interconnected and interdependent network of servers, PCs and intelligent devices, the meaning of "secure software" has evolved in turn. Once largely a matter of physically securing the machines on which the software ran, security now encompasses a wide range of practices, affecting everything from the source code itself, to its implementation, to the policies and standards that govern how and where that code is used. And as more business processes rely on the public Internet, security has become a deciding factor for a far wider range of companies seeking to purchase software, many of whom haven't needed to consider security issues in the past.

[3] Steve Lohr, Go To, Basic Books, 2001, page 204-205.

As such, security is now a tremendously complex and wide-reaching issue. Ensuring a secure computing environment requires different strategies at different levels; large organizations, mid-sized businesses and individual users all have different and unique security requirements. Large companies must contend with massive and diverse computing environments that encompass a mix of different software platforms, commercial and custom-built applications and services – all accessed by an expanding user base with an ever-increasing array of devices. Individual end users, once worried only about crashed software and lost floppy disks, now face greater risk of everything from malicious code to fraud and identity theft. These evolving threats have one thing in common: only dedicated IT professionals had to worry about security in the past – today, it's everyone's problem.

The various "fronts" in the battle for security can be outlined as follows:

- *Technology:* The software itself, whether an operating system, library or development tool, stand-alone application or network service, must be fundamentally secure.[4]
- *Implementation:* The software must be implemented and used in a way that is resistant to attack and misuse, while maintaining the ability to further update and modify it as threat models evolve.
- *Policy:* Strong policies must exist, both at the organizational and governmental levels, to govern how software is used and to deal appropriately with individuals and organizations that maliciously exploit or misuse the software.
- *Procedures:* Software providers must have mechanisms in place to accept customer feedback and respond to security issues, while organizations and individuals must be able to effectively implement needed changes.
- *People:* While technology is capable of ensuring a very high level of security, its users must still be aware of, and vigilant against, potential issues.

Source Access and Security

An important question is being asked of the commercial software industry today: does broad access to source code under the OSS model *inherently* lead to more secure software? Microsoft's position is that it does not. While source code review can result in some real benefits for security, experience has shown that the OSS model alone does not itself result in effective security review.[5]

[4] Computer security typically refers to the prevention of unauthorized disclosure of information (confidentiality); prevention of unauthorized modification of information (integrity); and prevention of unauthorized withholding of information or resources (availability). See generally Dieter Gollman Computer Security, John Wiley & Sons, 1999.

[5] "Open-sourcing your software can help improve your security, but there are associated risks to understand and consider if you're going to see real benefits. Open-sourcing your software is not a cure-all; it's an effective supplement to solid development practices and pre-release source-level audits." John Viega, "Open Source Software: Will it make me more secure?", September 1999 http://www.ibm.com/developerworks/library/oss-security.html (visited June 2002)

Thus, the complexities related to software security lead to a net-neutral result. Security issues in software are an industry-wide problem, and flaws are no more or less prevalent in software produced under either of the models.[6] Due to the popularity of Microsoft products, and the extensive use of our technologies for business-critical functions, the press tends to report more broadly on security vulnerabilities in our products. However, a balanced comparison of the number and severity of security vulnerabilities in commercial and open source software would show rough parity between the two types of software.[7]

If anyone can view and evaluate source code, find security flaws and fix them, some argue that this "many eyes" approach[8] is vastly superior to models in which source code is not broadly disclosed and the responsibility for fixing code lies with the company that produced it.[9] Moreover, many advocates argue that security is further improved by the pride that open source programmers feel in writing good code and the lack of commercial pressure to "do it fast" rather than "do it right." In fact, commercial software companies are under much greater pressure to "do it right," as more and more customers view security as a key decision factor when choosing a software platform.[10]

Development models ultimately have a neutral effect on software security. Every development model plays an important role in creating a healthy software industry, and each has its benefits and drawbacks. Software security is a complex and multifaceted issue where no single development model is the sole determinant of the quality of the security in a given package of software.

There are a number of factors that impact security beyond the development model used to create software. Those factors, and Microsoft's position on the role they play, are outlined below:

---

[6] From May 2001 through May 2002 Microsoft reported 60 security vulnerabilities for Windows and Security Focus Inc. reported 68 Linux security vulnerabilities. Microsoft Security Response Center: (http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/current.asp) Security Focus Inc. Bugtraq online (http://online.securityfocus.com/archive/103/2002-05-11/2002-05-17/0)

[7] This conclusion is drawn from the comparison made in the previous footnote. The counting of vulnerabilities becomes far more complex as you begin to look across multiple technologies from multiple vendors. For example, the 60 reported vulnerabilities for Windows include the operating system components of the Internet Explorer and Internet Information Server. There are related OSS technologies carrying their own security vulnerabilities (such as Apache and PHP) that were not included in the count of 68 Linux vulnerabilities.

[8] The "many eyes" theory breaks down upon closer examination of open source projects. The top 100 mature open source projects on SourceForge.com (the most popular OSS project foundry) have an average of 4 developers. There are only a few OSS projects that have attracted truly broad-based attention such as Linux and Apache. Cave or Community? An Empirical Examination of 100 Mature Open Source Projects, May 2002, http://firstmonday.org/issues/issue7_6/krishnamurthy/

[9] The many eyes theory was also questioned in a recent CNET article; Building trust into open source, CNET News.com, Robert Lemos, March 20, 2002. "'It does not assure that many eyes are actually looking at the code," Cowan said, "In fact, it is likely that 'rock star' code that is hip to work on gets adequate attention, while the other 90 percent languishes, in many cases never even seen by anyone but the code's authors." And much of this unsexy code forms the foundation of Linux."

[10] See Steve Lipner and Gary McGraw, "Open Source vs. Commercial Software: Which is more secure?," July 16, 2001, http://www.netmon.com.hk/a8.html (visited June 18, 2002)

Accountability and Trust

As more and more aspects of society become dependent on computing, customers will increasingly demand dependably secure software – and companies that fail to deliver on this requirement will soon find themselves without any customers. Commercial software companies therefore have a very strong incentive to ensure the greatest possible security, and they must be willing to invest whatever resources are necessary to do so.

Open source development models, on the other hand, have a more diffuse sense of accountability – it is unclear, from project to project, whether the responsibility for security lies with the collective of programmers who contribute to the code, the service providers who help customers implement and maintain the software, or the customers themselves. If the result is that security falls to either the service provider or the customer, the long term cost of that software dramatically increases.[11] It is even more difficult to determine, in the event of a flaw, who has ultimate responsibility for the code. As complex software drives more and more critical functions, customers will increasingly need an identifiable entity that can stand behind the software they use.

Visibility of Source Code Vulnerabilities

While the free availability of source code makes it easier for a large number of programmers to evaluate the code, and certain licensing approaches make it possible for them to develop and implement their own fixes, this availability does not guarantee that these activities will occur or that the results will be more secure. Experience shows that developers use their access to source code to change it rather than review its security, and users use source code access to build and maintain their systems.[12] Repeated discoveries of security vulnerabilities in shared open source components such as Sendmail[13], WU-FTPD[14], and Bind[15] illustrate the fact that open source software is anything but vulnerability-free. The result is that security of the source in question then depends as much on the quality of the security response mechanisms supporting that code as on the underlying quality of that code as it is implemented.

It is worth noting that the licenses most commonly associated with OSS mandate that the source code be available to all parties.[16] Thus, there is no ability to discriminate between

---

[11] Sea generally Calculating the Total Cost of Development, RSA Security Inc., February 2002, http://www.rsasecurity.com/solutions/developers/whitepapers.html

[12] "People using open source programs are most likely to look at the source code when they notice something they'd like to change. Unfortunately, that doesn't mean the program gets free security audits by people good at such things. It gets eyeballs looking at the parts of the code they want to change." The Myth of Open Source Security, John Viega, http://www.earthweb.com/article/0,,10455_626641_1,00.html

[13] Discussed in Ross J. Anderson , Security Engineering: A guide to building dependable distributed systems, John Wiley & Sons, 2001, p. 536

[14] See http://www.wu-ftpd.org, which provides a complete view of the CERT advisories related to the wu-FTPD security vulnerability

[15] See "Researchers Find Software Flaw Giving Hackers Key to Web Sites," Wall Street Journal(January 30, 2001); for the relevant CERT advisory, see http://www.cert.org/advisories/CA-2001-02.html)

[16] The Open Source Definition version 1.9, Open Source Initiative, http://www.opensource.org/docs/definition.php

trusted and un-trusted individuals or organizations. The broad availability of source code becomes a double-edged sword where the community may provide many eyes evaluating the source for security vulnerability, but it also provides a well-resourced and motivated party the means to conceive increasingly sophisticated attacks. It is for this reason that many commercial software companies choose to share source code under programs where large numbers of trusted parties may evaluate the security of the source code while avoiding the potential risks associated with unlimited distribution.[17]

Microsoft does not advocate "security through obscurity" – instead, we believe that it is possible to realize the benefits claimed for a "many eyes" approach to security by working with as many motivated and capable "eyes" as possible while reducing risks by limiting that community to trusted entities.

Quality of Source Code Contributions

The ability for any programmer to contribute code to open source software does widen the potential for someone to find solutions for security issues. However, this approach provides no assurance of the quality – or accountability – of fixes that may come from programmers with a wide range of expertise and commitment. Furthermore, there is evidence showing a high cost of both time and money for organizations seeking to bring security development in-house using open source software.[18]

Unfortunately there is a general shortage of security development experts in the software industry today. It is equally hard for commercial and community-based projects to take advantage of these highly-specialized individuals. The result is that the true security experts are going to become very expensive resources. With the increased public and industry focus on security, commercial software companies have a clear incentive to invest in highly trained employees to improve the security in their products.

Testing and Implementation

A complete evaluation of source code for real or potential security vulnerabilities is an incredibly difficult process, especially for large, complex software such as an operating system. Moreover, an increasingly networked and interdependent computing environment requires further degrees of testing, and larger environments (such as those in large companies) require specialized testing and implementation – and this burden will only increase as Web Services grow in importance. Rapid, thorough regression testing of even the smallest code changes takes tremendous time and expense. Commercial software companies, who are ultimately responsible and accountable for their products, have a clear incentive to perform this testing. On the other hand, community-driven projects

[17] The Microsoft Shared Source Initiative has extended source level access for Windows 2000, XP and .NET Server betas to more than 2000 customers, partners and governments in 30 countries. Many of the organizations who have Windows source code, such as the Austrian government, have chosen to license the source code for security reasons.

[18] See Calculating the Total Cost of Development, RSA Security Inc., February 2002, pages 1-2 http://www.rsasecurity.com/solutions/developers/whitepapers.html

with more of an "ad hoc" approach to creating, testing, distributing and publicizing security patches tend not to have the same quality control capabilities.[19] The result is that more of the burden and expense of integrating those patches fall to the individual organizations using the software.[20]

Government Certification

Over the past decade, government information assurance programs have taken on even greater importance, as more companies look to governments for guidance and the governments themselves make purchasing decisions based on these criteria. Throughout the 1990s, a number of security evaluation programs from the United States, Canada and Europe came together to form a guideline known as the Common Criteria[21], which is now adopted by the governments of fifteen countries.[22] These guidelines carry significant requirements for documentation, evaluation, design and test reviews – a rigorous and expensive process for anyone who creates software.[23] Commercial software vendors have a clear incentive to satisfy their government customers, and are thus devoting tremendous resources to attaining this certification.

The effort required to acquire government certification such as the Common Criteria may go beyond the logistical and monetary capabilities of a community-driven project. Even if a large commercial entity were to assume the responsibility and costs for certifying an OSS project, the path to certification would be especially arduous due to the distributed nature of code contributions in OSS projects and the lack of uniform documentation and testing processes across all contributing developers and subcomponents. The task would be made even more difficult given the fact that the certification process must include any additional features and subcomponents that are added to the evaluated configurations as the product continues to mature and responds to customer and competitive demands. This is a difficult task to manage unless the certification process is handled by skilled professionals.[24]

[19] Open Source vs. Commercial Software: Which is more secure?, Lipner & McGraw, July 2001. The wu-FTP open source code had undergone security reviews for years prior to 6 critical flaws were found in 2000. Another example is the MIT Kerberos V5, a security technology, was in circulation for 10 years with undiscovered buffer overflows due to the fact that everyone assumed someone else was looking at the source code.

[20] Calculating the Total Cost of Development, RSA Security Inc., February 2002, http://www.rsasecurity.com/solutions/developers/whitepapers.html

[21] Common Criteria, http://www.commoncriteria.org

[22] The current recognized participants of the Common Criteria standard are Australia, New Zealand, Canada, Finland, France, Germany, Greece, Israel, Italy, Netherlands, Norway, Spain, Sweden, United Kingdom and the United States. http://www.commoncriteria.org/site_index.html.

[23] It is unclear that projects being built under the open source software model will be able to meet the requirements of documentation and testing as laid out in the Common Criteria guidelines.

[24] If a vendor is neither the original producer of the software, nor the rights holder of the software they may be extremely reluctant to put themselves in the position of representing the fitness of that work to meet the Common Criteria guidelines.

Response Mechanisms

As the computing environment has evolved, the manner in which commercial software companies have responded to security issues has evolved in turn. Early in the software industry, many companies – whether through arrogance or ignorance – dismissed many security vulnerabilities, viewing them as un-exploitable or considering their exploitation to be very unlikely. As such, their response to security issues was largely defensive, slow and inadequate. As security became a greater concern, companies began to see vulnerabilities as inevitable, and developed formal processes to discover, evaluate and fix them when they arose.

This approach created a new challenge – as these response mechanisms matured, organizations found themselves increasingly swamped by a confusing array of patches with limited ability to determine the severity of one vulnerability over another. Thus the decision process associated with determining which patches to deploy became increasingly difficult. Not only did this make maintaining large computing environments more challenging, it also drastically raised the visibility of security issues. Success therefore depends not only on how many vulnerabilities are identified and fixed, but how many of these fixes are actually installed and used.

Today, commercial software companies are increasingly focusing on security as a "holistic" concern. They recognize that security depends on thorough protection of everything in the computing environment from every possible security threat, and thus have focused on building security into their products from the ground up. They are writing better code, testing it thoroughly, then offering tools and information that help their customers manage the ongoing process of ensuring security. Again, since their success depends on reaching this level of security, commercial software companies have a tremendous incentive to take action.

Microsoft's Shared Source Philosophy

At Microsoft, we believe the popularity of open source development models has brought a healthy level of competition to the software industry, and will continue to drive debate in the industry.

The company recognizes the benefits and drawbacks of source code availability and public contribution to source code, and their potential impact on security. We continue to expand the programs under which source code for our products is available to customers, partners and the academic community.

As a result, Microsoft source code is now being viewed by tens of thousands of organizations around the world – Windows source code is available for review in over 30 countries, giving governments, customers and partners who want a deeper understanding of the security elements of our products the ability to do source-level reviews.

The philosophy behind these programs – which Microsoft calls “Shared Source” – rests on four key principles: first, to empower customers and the developer community to be successful through source code access programs and powerful development tools; second, to incorporate more customer feedback into products; third, to nurture the continued development of a robust and healthy software industry by expanding source access to universities and academic institutions worldwide; and fourth, to protect intellectual property rights through source licensing policies tailored to different customer and developer needs while preserving the economic incentive to invest in further research and development. This balanced approach allows Microsoft to make its source code more widely available in a manner that is consistent with the creation and maintenance of a sustainable business model.

Ultimately Microsoft and all other producers of software contribute to a cycle of sustained innovation based on strong intellectual property rights and a commitment to ongoing investment in research and development. The responsibility for the creation of secure software falls to all contributors in the software ecosystem independent of the development model utilized in the creation of software.