Open Source and Assurance

Ian Tien, September 10, 2002

The debate regarding the security of open source software is long standing. Some would say “no, open source does not provide sufficient assurance for important projects” and highlight the fact that secure systems are almost exclusively created from non-open source code. Others would say “yes, it works” and point out the thousands of widely used and generally secure open source software installations such as FreeBSD, Apache, Linux, and Sendmail [9].

So the concise answer to the debate is: “yes, there are instances where open source software offers sufficient assurance, however there are many cases in which it does not.”

The major barrier to the security of open source systems is the lack of infrastructure. As system security is a holistic art, without the automated tools to rigorously test huge volumes of complex source code, the assurance provided by an open source system is limited by the cognitive abilities of its many developers.

Nonetheless, the open source effort has recently achieved a number of significant milestones, such as DARPA’s CHATS program supporting open source development and the NSA’s release of “Security Enhanced Linux” [10].

The security in open source software is a hotly debated topic and factual arguments are often mixed with misinformation and rhetoric from the stakeholders of the debate [1, 2, 7, 9, 10]. An exhaustive rebuttal of all the “issues” raised on the topic is therefore beyond the scope of this article.

Instead, we present an overview of key arguments for and against open source software from the perspective of providing assurance to software customers.

Bad: There is no accountability for open source
It has been argued that institutions "cannot afford to use hapless and untested software without accountability, warranties or liability" [2] and that many government departments are ill-equipped to support security issues in their software. The counter argument is that that there has been a fundamental shift in the software industry from ownership to service [7] and that there exist companies that will offer quality support and maintenance equivalent to that of a proprietary software company for open source projects. As an example, the consulting arm of IBM, the world's largest software company, is willing to take on contracts to provide solutions that depend on the security of open source systems such as Linux.

Good: Open source offers economies of scale
The overhead required to create software from scratch can be alleviated by the use of ready-made software. Firms using open source software are therefore able to spend more time on testing and analysis and less time on rebuilding systems that already exists. The end result is a more secure system for the same amount of resource investment. A similar argument can be made for proprietary software sold at a reasonable cost by third party vendors.

Bad: Proprietary firms catch and fix bugs faster
It has been proposed that security holes in proprietary software can be caught and fixed more quickly than in open source systems, which depend on volunteer networks [1, 2]. However the same argument can be made that catching and fixing holes in open source would be faster since the source code of all components is available and can be readily modified.

Bad: Open source efforts lack infrastructure
The cathedral of proprietary design often comes with architects, analysis tools and automated testing environments that provide resources with which the open source community cannot easily compete.

Bad: No system is invulnerable
No software system is invulnerable [3, 4, 5, 6, 8]. However it has been argued that open source is more vulnerable since attackers can setup test environments to practice their attacks [2].

Bad: Some vulnerabilities cannot be defended against
There may exist vulnerabilities in software systems that cannot, or cannot economically, be defended and which must rely on security by obscurity [8]. In an open source system no protection would exist.

Good: Open source allows for peer review
The behavior of an open source system is constantly under peer review by open source contributors and the probability of unnoticed errors decreases with the number of qualified reviewers reading through the source code. However, this benefit works only under the assumption that there are a sufficient number of interested eyes.

Bad: Open source is subject to subversion
One argument against open source efforts is that they are susceptible to the injection of bugs and weaknesses by malicious or unskilled developers. The counter argument is that proprietary firms can be equally malicious and unskilled. The hidden flight simulator

game in Microsoft Excel and the pinball game in Microsoft Word serve as examples of such undetected functionality [10].

Good: Open source code can be fixed
The comment has been made that "depending on code without the source is quite similar to depending on a complex mechanical or electronic system without the benefit of shop and parts manuals…" It is true that holes in open source systems may be faster to repair, however they are also easier to exploit.

Bad: Open source systems may lack support
Open source projects can fall apart if the project community loses interest. At the same rate there is no guarantee a proprietary firm will continue to support its products. Such firms may go into a new line of business, be under financial distress or even go bankrupt.

Bad: Open source systems are poorly documented
Open source systems often have insufficient or incomplete documentation, since writing documentation is not often considered to be an enjoyable pastime. However most widely used open source systems are well documented and may even feature published documentation from companies such as O'Rielly books.

Bad: GPL requires disclosure of modified source code
This is true only if the system is distributed.

Given the pros and cons of both open source and proprietary approaches, it is clear that the "better" solution is largely dependent on a wide range of factors, such as the level of assurance required, the development resources available and the functionality of the desired system.

---

[1] Broersma, Matthew (2002) Study: Open source poses security risks.
http://zdnet.com.com/2100-1104-929669.html

[2] Brown, Kenneth (2002) Opening the Open Source Debate, Alexis de Tocqueville Institution. http://www.adti.net/cgi-local/SoftCart.100.exe/online-store/scstore/p-brown_1.html?L+scstore+llfs8476ff0a810a+1042027622

[3] Charles P. Pfleeger. Security in Computing. Prentice Hall PTR, New Jersey, 1996.
Bruce Schneier. Applied Cryptography. Second Edition. Wiley, 1996.

[4] Charlie Kaufman, Radia Perlman, and Mike Speciner. Network Security. Private Communication in a Public World. Prentice Hall, 1995.

[5] Committee on Information Systems Trustworthiness, National Research Council (1999) Trust in Cyberspace.
http://www.nap.edu/readingroom/books/trust/

[6] Dieter Gollmann. Computer Security. Wiley, 1999.

[7] Duartenn, Juilao (2002) Comments on "Opening the Open Source Debate" http://www.juliao.org/pub/adti-comments.pdf

[8] Fred B. Schneider, Lynette I. Millett and Jed Liu (2002?) 513 Course Notes
http://www.cs.cornell.edu/Courses/CS513/2002FA/L01.html

[9] Pavlicek, Russell (2002) Shred that paper, InfoWorld:
http://www.infoworld.com/articles/op/xml/02/06/24/020624opsource.xml

[10] Viega, John and Fleck, Bob (2002) Dispelling Myths about the GPL and Free Software, Cyberspace Policy Institute.
http://www.cpi.seas.gwu.edu/oss/oss.html