# The security implications of open source software



## Does open source mean an open door?

Natalie Walker Whitlock (Natalie@Casaflora.cc)
Casaflora Communications
March 2001

Natalie Whitlock talks about the incongruence of closed security systems, and the open source solution. She discusses Eric Raymond's ideas, the famous "back door" in Microsoft's FrontPage, the concept of peer review, and the open source dilemma that no one is at the helm guaranteeing that everything will be checked. She then follows the idea from theory to practice and talks with leading IT executives about the viability and popularity of secure open source systems.



Contents:



Related dW content:

"Open source software: Will it make me secure?"

To some, closed source means hidden, secret -- and more secure. In reality, many of the most secure systems available today are based on the open source model.

Traditionally, secrecy has meant security. You lock up your house, your automobile, your valuables. In the software community, you "lock up" the programming source code as a means of securing it against hackers and competitors. To the closed source camp, a system can't be truly secure when its source is open for all to read. Secrecy is security, and when applied to an otherwise secure system, concealing the source improves the security. It slows up intruders and, in the event of a breech, keeps damages at a minimum. Another argument is that with freely available blueprints, crackers will have it easy writing malicious code to attack systems.

### The security of knowing it's open source

What, then, about the security of open source software? Open source software, by definition, is any program or application that is freely distributed, non-platform specific -- and in which the programming code is open and visible. All else being equal, isn't a closed program more secure than an open one?

"Not just no, but h--- no," says open source advocate and author of *The New Hacker's Dictionary* and "The Cathedral and the Bazaar", Eric S. Raymond. "Closed source leads not to true security but to a false sense of security," says Raymond. "You don't know what's in there, you can't verify it, you can't check the assumptions or honesty of the people who wrote it."

In cryptography circles, there is a saying: The security of an algorithm should not depend on its secrecy. This maxim can be especially well-applied to open source software.

"The apparent paradox that openness about your methods leads to better security is not unique to computer software -- military and diplomatic cryptographers have known for a century that it is folly to depend on the secrecy of your encoding method rather than the secrecy of your keys," says Raymond.

According to Raymond and fellow open source supporters, open source is the only real option for secure

operating systems. For one thing, closed source applications and operating systems can't be examined and verified for secure coding. A revelation of previously secret code almost always leads to the discovery of additional flaws and security holes. In addition, closed proprietary code makes it difficult to distribute trustworthy fixes when a hole or mistake is revealed.

Just take the April 2000 event that had webmasters and systems administrators shaking in their shoes. After four years it was discovered that Microsoft programmers had inserted a back door in their popular FrontPage Web server software. It was the very fact that the software code was "concealed" in opaque binary form that kept this security breach unknown to the public for so long.

Since open source software guarantees the "right to read, redistribute, modify, and use the software freely," a secret back door would be highly unlikely to escape detection. Most experts believe that the odds of a such a back door slipping in are nil. After all, the logic goes, who would risk his or her reputation by putting a back door in source that is openly available for others to discover?

"Anybody who trusts their security to closed-source software is begging to have a back door slipped on to their system," says Raymond, who is also president of the Open Source Initiative. "Apache has never had an exploit like this, and never will. Nor will Linux, or the BIND library, or Perl, or any of the other open source core software available."

**Many eyeballs**

The tacit security of open source software comes from the concept of "peer review," borrowed from the scientific community. According to OpenSource.org, the official Web presence of the Open Source Initiative (OSI), open source software "promotes software reliability and quality by supporting independent peer review and rapid evolution of source code."

The basic Linux security philosophy can be summed up as "more eyes means better security." In other words, since the source code is open for all to review, numerous programmers worldwide will be examining the code for security vulnerabilities. On the other hand, closed source applications will only be audited for vulnerabilities by paid programmers working behind closed doors on proprietary products or trade secrets -- or by system crackers.

"Closed source means only the bad guys get to find the security holes. With open source, good guys get to look for them too," says Raymond.

This promise of peer review promotes secure coding techniques among open source projects, argues Raymond. "Even the mere prospect that the source code will be reviewed by skeptical peers changes the behavior of developers," he says. "They write tighter, better, more error-free systems."

"In an open source project, to make a mistake and have it known to the entire development community and your friends is mortifying in the extreme," says Michael H. Warfield, senior researcher with Internet Security Systems Inc. and UNIX systems engineer and consultant. "That last moment before hitting the Enter key -- to commit a change or send a patch out into the cold cruel world of your peers -- is the longest moment imaginable."

Warfield, who has worked in both open and closed source development models, says there is a higher standard of personal accountability and greater professionalism among open source programmers. "When the source and changes to it are present in public for anyone to examine, it becomes personally incumbent on the developer to ensure the code is right and that it hasn't been tampered with by any unknown parties," says Warfield.

And what happens when security issues do crop up in open source? "They get fixed. Rapidly," states Raymond. Just contrast the Microsoft incident with the "Ping 'O Death" bug, an attack based on firing off oversized ICMP packets, which was fixed in Linux within only a few hours after it was announced. Because of peer review, open source problems can generally be found and fixed before they're widely exploited.

Another perk of open source is that the software actually evolves and gets more secure over time. Subject to constant peer review, the number of new vulnerabilities discovered in the software will decrease over time

when compared to similar closed source software. But as more crackers seek and find the better-hidden flaws in opaque programs, closed source software gets less secure as time passes.

**Theory vs. practice**

Still, there are skeptics. "Simply being open source is no guarantee of security," says Craig Willis, computer consultant and a state government systems analyst. "Just as the good guys are looking for vulnerabilities, the same thing goes for the bad guys. 'Security through obscurity' may not be something you should depend on, but it can work to your advantage if the attacker can find an easier target."

Lee Badger, principal computer scientist at Network Associates, agrees, countering that the many-eyes theory "assumes people are motivated to examine even the mundane code -- and I'm not sure that's the case."

Even a few open source advocates admit the dilemma. The author of the GNU mailing list manager Mailman, John Viega, revealed that for three years Mailman had a handful of glaring security code problems, yet no one caught or reported the errors. The version of Mailman that contains these security holes was downloaded thousands of times and even included in Red Hat Professional Linux version 6.2 until mid-2000. Apparently, says Viega, everyone using Mailman assumed that someone else had done the proper security auditing, when in fact, no one had.

Says Viega, "The benefits open source provides in terms of security are vastly overrated, because there isn't as much high-quality auditing as people believe, and because many security problems are much more difficult to find than people realize."

Viega points out that even after they were identified, the security problems in Mailman took many months to fix, partly because it was written in Python (rather than the more popular C) and partly because security was not the core development team's immediate concern.

Viega lists several things that can discourage people from reviewing source code, including code that looks "like a tangled mess," or programs written in a lesser-used language. Another deterrent is human nature, as most programmers look at source code for their own benefit, not for altruistic motivations.

One issue some detractors cite is that open source code is only as good as the skill of those who review it. "Many (developers) don't understand enough to avoid problems beyond the handful of dangerous calls they know," says Viega. According to his article on open source myths (see Resources later in this article), it is common for developers to use cryptography, but misapply it in ways that destroy the security of the system, or to use encryption that is too weak and can easily be broken. Another mistake is for developers to try to hand roll their own protocols using common cryptographic primitives, not fully understanding that cryptographic protocols are generally more complex than expected and easy to get wrong.

Theo de Raadt, project leader for the open source operating system OpenBSD, is another who doubts the safety net of peer review. "These open source eyes that people are talking about -- who are they?" de Raadt asks. "Most reviewers of open source code are amateurs. Most of them, if you asked them to send you some code they had written, the most they could do is 300 lines long," he says. "They're not programmers."

Another complaint with open source software is that most bugs are found after the program has been compiled, tested, and distributed -- not because someone sat down in advance and looked at the code for holes, but because something goes wrong in its use. With a few exceptions, open source programs do generally rely on user reports and public forums to find vulnerabilities.

**The beat goes on**

Despite the holdouts, many throughout the industry are embracing the open source model. In fact, an August 2000 Forrester Research report forecasts that all traditional software vendors will need to change their proprietary business model to an open source one within the coming four years.

In the study, Forrester found that out of the 2,500 IT managers interviewed, 56 percent currently use open source software and another 6 percent plan on installing it in the next two years. A full 84 percent of the interviewees agreed that open source software will spark major innovations throughout the industry.

Most importantly, the study reported that "security issues" was the most often-cited reason to adopt open source software.

"No surprise to me there," says Piers Van Dorf, a London-based security consultant. "If you want to secure your system," says Van Dorf, "open source is the way to go. When you have a lot of impartial people really looking at the source code, mistakes -- intentional or otherwise -- have less of a chance of sneaking through. You won't keep everyone out all of the time, but you will keep most of the people out most of the time."

Finally, keeping source code "closed" doesn't necessarily equate to hiding it. Just because a hacker can't see the original programming doesn't mean he can't run it in a code execution trace environment or through reverse engineering. Hackers have been profiting by "Black Box analysis" for years. There are even tools out there now to automate the Black Box analysis process. Given that any code that can run can be disassembled, it stands to reason that keeping it obscured serves no valid security purpose.

What will be the final word in the "many eyes" vs. "security by obscurity" controversy?

"I don't think the security community has come to terms with this open source thing yet," says Avi Rubin, AT&T researcher and author of *The Web Security Source Book*. "This is the next great debate."

**Resources**

- Read John Viega's article "Open source software: Will it make me secure?" in *developerWorks'* Security special topic.
- Follow the latest security news at SecurityFocus.com's BugTraq.
- Understand the ins and outs of OSS at the Open Source Initiative Web site.
- Visit Eric S. Raymond's home page.
- Read the Ping o' Death Diary.

**About the author**

Natalie Walker Whitlock is a freelance writer and the owner of Casaflora Communications, a content service specializing in e-business and technology issues. She has written for numerous print and online publications including *PC World*, *Office.com*, *iVillage*, *Intraware*, *The Tribune Syndicate*, *The Arizona Republic*, *PC Parents*, and *CBS Marketwatch*. You can contact her at Natalie@Casaflora.cc.