WRITTEN TESTIMONY OF BRIAN BEHLENDORF.
CHIEF TECHNOLOGY OFFICER
COLLABNET
BEFORE THE COMMITTEE ON HOUSE ADMINISTRATION, ELECTIONS SUBCOMMITTEE
U.S. HOUSE OF REPRESENTATIVES
MARCH 15, 2007

Members of the Committee, good afternoon and thank you for the opportunity to speak to you this afternoon on the topic of "disclosed" and Open Source software. My name is Brian Behlendorf. I am the Founder of CollabNet, a global company focused on providing tools and services for collaborative software engineering, as best exemplified by the Open Source software community. For the last 8 years I have served as its Chief Technology Officer and Executive Board member.

Today I'm going to talk about how Open Source is the continuation of a series of transformations that have taken place in the software industry. I'll explain how it has become a dependable mechanism for software development and commerce, how it can lead to more secure and trustable software, and how it serves the interests of the customers by reducing vendor lock-in. I'll also explain the real differences between Open Source and simply "disclosed" software, an understanding that is critical as you look at the language of proposed legislation.

To further explain my context for these comments, I was a co-founder and the first President of the Apache Software Foundation, a U.S.-based 501c3 nonprofit organization responsible for the technology used in over 60% of the web sites in existence today. Currently I serve on the Executive Board of the Mozilla Foundation, the organization responsible for the Firefox web browser. I also served for three years on the initial Executive Board of the Open Source Initiative, the organization responsible for the defining the "Open Source" trademark and educating the public on the concept. I speak today on my own behalf.

The software industry has seen a sequence of deep and often disruptive transformations throughout its brief history, with each transformation creating new opportunities and new industry leaders. The first major transformation, in the 1970s and 1980s, was called "Open Systems", which promoted the unorthodox notion that software should be built that could run on different kinds of hardware. Microsoft was born during this transformation and profited tremendously from its premise, as did many other software companies we know of today. Some other companies, such as IBM, adapted to the changing environment, survived, and thrived. Others resisted the move, and perished.

The second major transformation, which came to prominence in the early 1990s and yet is still underway, was that of "Open Standards". The unorthodox notion this time was that two companies, ten companies, or more could meet as peers and create common vocabularies for interchanging data between different pieces of software. The greater the number of software programs that used this common vocabulary, the greater the total amount of value created. From this concept was born the Internet, the network of networks, made possible only by the principle of sharing a common network vocabulary (called TCP/IP) as widely as possible. As with the first transformation, we saw new companies like Cisco and Sun emerge, we saw other existing companies adapt and thrive, and we saw others resist and perish.

The third major transformation to have taken place in the software industry is that of "Open Source" software. Open Source software is software licensed under a generous copyright license; licenses that allow many kind of use at zero price, that provide access to the underlying application "source code", that allow modification and improvement, and that allow the recipient the right to share their modifications with others. Here, the unorthodox notion is that this approach can result in fewer defects, greater flexibility, more rapid innovation, more responsive vendors, and a more competitive marketplace than the more proprietary alternatives.

Today, every major technology vendor releases some portion of its intellectual property under an Open Source license. The business models these companies pursue to justify such an investment are a mixture of support, services, and strategic opportunities created for other proprietary offerings. Red Hat is the most famous example of this, commanding a market capitalization of over $4B. Traditional technology companies have embraced this too: Sun, HP, and IBM all have significant revenue streams based on Open Source software. Even Microsoft has acknowledged some value to this approach, not just by partnering with Novell to co-sell Linux to Microsoft customers, but by also releasing some small Open Source projects themselves.

On the customer side, Open Source software has crossed the chasm from its early adopter support amongst the engineers to enterprise production use. Every firm on Wall Street I have talked to depends upon Linux and other Open Source software to execute trades or conduct other financial transactions. Many consumer devices invisibly embed Open Source technologies, from cell phones to Tivos to automobile electronics. Within the public sector, the use of Open Source software has grown tremendously, in such demanding agencies as the Pentagon, Commerce, Energy, and Homeland Security. In all these environments, Open Source software and proprietary software can co-exist, thanks to open standards and open systems.

Is Open Source software guaranteed to be more secure? In software, as anywhere else, there are no guarantees. It is extremely challenging for even the most competent engineers to write invulnerable code - it's as likely as planting and managing a garden that has no weeds. New methods of attack are discovered all the time, and the re-use of software in new settings can often open new holes. Yet the ability to prevent mistakes or external compromise in certain situations, such as electronic voting systems, is critical.

The only widely recognized indisputable method to achieve low-defect software is developer peer review. Eric Raymond, the author of The Cathedral and the Bazaar, a paper that first popularized the concepts around Open Source software, once said, "To enough eyeballs, all bugs are shallow." The more widely inspected code is, the lesser the chance of the undiscovered defect. This extends to the development process itself - the larger the development team around a given body of code, and the more that the deliberations of that team are opened to the outside world, the more reliable their designs are likely to be. This "community" approach is the key ingredient to any successful Open Source project.

An illustration of this is the OpenSSL project. Launched over 12 years ago, this is a library of cryptographic routines and tools and functionality that is used to secure everything from credit card and other sensitive transactions over the Internet, to "smart cards" for accessing physical systems. This library has become the reference platform for building cryptographically secure applications. Written by individuals working around the world, this library has received extensive scrutiny from security professionals and researchers worldwide, and has gained FIPS-140 certification for use in U.S. government applications. Like any piece of software, there are bugs, and occasionally one is found and reported to the development team. Rather than deny the existence of such a bug, the public nature of the project forces them to embrace that discovery, fix it as quickly as possible, and issue an update - often within a matter of hours, almost always within a few days. This level of scrutiny, and the degree of responsiveness, has built confidence in the hearts and minds of security professionals everywhere in OpenSSL.

If this were a commercial product forced merely to "disclose" its source code with carefully selected partners in a closed manner, the chances of a community forming to review that work effectively and sufficiently to gain that same level of trust, are close to zero. This is why the security and effectiveness of an "Open Source" system is not merely about "disclosure", but about co-development between peers, and the creation and promotion of common technologies to solve common problems.

Finally, the most useful aspect to choosing an Open Source product is the inherent protection it can give against vendor lock-in. A support customer of one Linux vendor, has the freedom to shift to another Linux vendor should they become dissatisfied with the first. The customer's investment of training time on Linux, improvements to Linux, and in technology on top of Linux, does not have to be thrown away should the commercial relationships change. Open Source allows the redefinition of the traditional relationship between customer and provider, from one of dependency towards one of enablement and cooperation.

Customers of vendors selling Open Source electronic voting software necessarily retain the legal rights to continue to use and improve the software, even if they elect to switch to another vendor. The vendors will continue to have a lucrative market to pursue - that of providing and maintaining the election hardware, the customization of the software to each precinct's needs, and providing support services before, during, and after the election. Such activities are complex enough to create plenty of opportunity for relative competitive advantage for each vendor. Further, each vendor's R&D costs would be reduced, as the development of common software is shared between multiple vendors, and could involve volunteers, non-profit organizations, or government-funded contributors. Viable Open Source software designed for voting systems already exist, and have been used in elections in Australia, though no such system has yet been deployed in the United States.

To summarize, the Open Source transformation taking place in the software industry today is real, it is pro-business and pro-customer, and it has a tremendous chance to build trust in the security and proper operation of such software. It alone can not guarantee a trustable electoral process, but in conjunction with other solutions it can play a key enabling role. And along the way, it can help redefine the relationship between the public sector and the system vendors in favor of the public interest.

Thank you again for allowing me to testify.

Brian Behlendorf