On Open Source Software

Nyimbi Odero

August 28, 2014

Abstract

This brief discussion paper makes the case for open source software in the IEBC. It has been argued, without rebuttal, that open source software is insecure, unreliable, more costly than commercial software and by implication unsuitable for serious business use. It will be demonstrated, with myriad examples, that exactly the converse is true and that the open source model of development promotes software reliability and quality by supporting independent peer review and rapid evolution of source code. The mechanics of open source software also lead directly to more secure software and services, lower costs for business and a longer operational lifespan for valuable software.

1 A Security Backgrounder

Information security, sometimes shortened to InfoSec, is the practice of defending information from unauthorized access, use, disclosure, disruption, modification, perusal, inspection, recording or destruction.[1] InfoSec typically concerns itself with two broad areas IT or Computer Security and Information Assurance.

- IT Security largely refers to the security of computer systems.
- Information Assurance is concerned with preventing the loss of valuable data in the face of emergent exigencies.

[1] Committee on National Security Systems: National Information Assurance (IA) Glossary, CNSS Instruction No. 4009, 26 April 2010.

1.1 Vulnerabilities

A computer system vulnerability is a weakness which allows an attacker to reduce a system's information assurance. The three tenets of cybersecurity define a vulnerability as the intersection of three elements:[2]

1. A system susceptibility or flaw,
2. Attacker access to the flaw, and
3. Attacker capability to exploit the flaw

To exploit a vulnerability, an attacker must have at least one applicable tool or technique that can connect to a system weakness. In this frame, vulnerability is also known as the ***attack surface***.

In order to reduce the attack surface and/or mitigate the cost of an attack in progress:

Shrink susceptibility: Focus on what's critical-

- Enumerating system access points and associated security elements
- Reducing access points to only those necessary to accomplish the mission

Restrict threat access: Move it 'Out of Band' - Make critical access points and associated security elements less accessible to an adversary e.g operate a closed network, provide physical protection of computer systems etc.

Deny threat capability: Detect, React, Adapt by

- Imposing appropriate penalties when attack is detected

[2] "The Three Tenents of Cyber Security". U.S. Air Force Software Protection Initiative (ATSPI). http://www.spi.dod.mil/tenets.htm

- React inside threat's OODA (Observe, Orient, Decide, Act) loop
- Fighting through the attack!

It should now be evident that the larger the attack surface of a system, the more vulnerable it is. Restricting access and denying threat capability are largely deployment issues. Shrinking the susceptibility of a software system is a function of design and flaws (bugs). Peer review has stood the scientific process in good stead since the renaissance, and is largely responsible for the tremendous technological growth that has characterized the modern world. Applying the same principles to software development has also led to fewer bugs in OSS. A study, for instance, of the Linux source code has found 0.17 bugs per 1000 lines of code while proprietary software generally scores 20–30 bugs per 1000 lines[3]. **Almost 200 times fewer bugs than comparable proprietary software**[4].

In the case of Linux, and more generally in the case of Open Source Software, the claim that OSS is less secure is debunked. It is noteworthy that the US NSA, ostensibly one of the most secure installations in the world uses Linux a version of Linux and a host of open source software[5].

2 Open Source

2.1 Development Model

In[6] his 1997 essay "The Cathedral and the Bazaar", open-source evangelist Eric S. Raymond suggests a model for developing Open Source Software (OSS) known as the bazaar model. Raymond likens the development of software by traditional methodologies to building a cathedral, "carefully crafted by individual wizards or small bands of mages working in splendid isolation". He suggests that all software should be developed using the bazaar style, which he described as "a great babbling bazaar of differing agendas and approaches."

In the traditional model of development, which he called the cathedral model, development takes place in a centralized way. Roles are clearly defined. Roles include people dedicated to designing (the architects), people responsible for managing the project, and people responsible for implementation. Traditional software engineering follows the cathedral model.

Generally OSS is developed by myriad people from across the globe, all of whom have access to the complete source code all the time. Changes, recommendations and bug-fixes are incorporated regularly (most commonly every night). There are normally two versions of the code or application available - a Stable release that has fewer features but has been extensively tested and a Development release that has all the latest features and code fixes but is not as fully tested.

2.2 Advantages

> **"We migrated key functions from Windows to Linux because we needed an operating system that was stable and reliable – one that would give us in-house control. So if we needed to patch, adjust, or adapt, we could."**[7] *Official statement of the United Space Alliance, which manages the computer systems for the International Space Station (ISS), regarding why they chose to switch from Windows to Debian Linux on the ISS*

Scholars Casson and Ryan have pointed out[8] several policy-based reasons for adoption of open source – in particular, the heightened value proposition from open source (when compared to most proprietary formats) in the following categories:

- Security: Open source systems are inherently more secure than proprietary systems. Like well run businesses public sector entities need to make security a primary concern when choosing a software vendor. Sensitive documents must not be compromised.

[3] http://archive.wired.com/software/coolapps/news/2004/12/66032

[4] http://web.stanford.edu/~engler/metrics-sosp-01.ps

[5] http://www.nsa.gov/research/selinux/

[6] http://en.wikipedia.org/wiki/Open-source_software (last visited Aug. 28, 2014).

[7] http://www.telegraph.co.uk/technology/news/10049444/International-Space-Station-to-boldly-go-with-Linux-over-Windows.html

[8] Casson, Tony and Ryan, Patrick S., Open Standards, Open Source Adoption in the Public Sector, and Their Relationship to Microsoft's Market Dominance (May 1, 2006). STANDARDS EDGE: UNIFIER OR DIVIDER?, Sherrie Bolin, ed., p. 87, Sheridan Books, 2006. Available at SSRN: http://ssrn.com/abstract=1656616

- Affordability: Companies come and go; public entities exist in perpetuity. For this reason, the concept of "affordability" is more important in the public sector, because poor governance and inefficient puchases shift the costs to future generations.
- Transparency: Public sector processes in open societies need to be open for inspection. Public sector institutions need to be fair, and their fairness must be transparent to the citizenry. Documents, as well as the processes used to create them, must be easily understandable to all interested members of the society.
- Perpetuity: Governments have an obligation to ensure that their records are preserved indefinitely. With increasing reliance on electronic formats maintenance of programs that cn decode these formats must be considered. The need to ensure perpetuity of access is a fundamental force behind the drive for open document standards and software.
- Interoperability: Public sector organizations may have dozens of applications and document formats. It is especially important that all software and documents can be understood and used across systems and functions.
- Flexibility: Unconditional access to source code results in current and future freedom to adapt the software to local needs.
- Localization—particularly in the context of local governments (who make software decisions). Casson and Ryan argue that "governments have an inherent responsibility and fiduciary duty to taxpayers" which includes the careful analysis of these factors when deciding to purchase proprietary software or implement an open-source option

3 Public Sector use of Open Source

There are myriad examples of entire governments switching to open source software examples include:

US: data.gov The US Data.gov is powered by two open source applications, CKAN and WordPress, and it is developed publicly on GitHub. Data.gov is the home of the US government's open data.

US: Government Every agency starting with the Department of Homeland Security use open source software[9].

govcode.org Lists over 1000 open source projects used by goverment over the last 12 months.

Brazil : Brazil[10] is a world leader in Open Source deployment in government globally, with a formal policy of using open source. For instance The Bank of Brazil, the largest public bank in Brazil, has more than 100,000 work stations, more than 6,000 servers, 15 IBM mainframes and more than 42,000 automated teller machine (ATM) terminals. Brazil has created laws to obligate entities, such as government departments, to use only OSS. In some departments, users are not allowed to use proprietary office software and must instead use OSS equivalents.

Most of Latin America : Entire countries have laws requiring the use of open source software. A complete list can be found at http://en.wikipedia.org/wiki/Adoption_of_free_and_open-source_software_by_public_institutions. Multiple German agencies rely largely on OSS. The whitehouse website is run on OSS.

[9] http://siliconangle.com/blog/2014/03/18/your-u-s-government-uses-open-source-software-oss-and-loves-it/

[10] http://timreview.ca/article/250