10TH INTERNATIONAL COMMAND AND CONTROL RESEARCH AND TECHNOLOGY SYMPOSIUM
THE FUTURE OF C2
13-16 June 2005
MacLean, Virginia, USA

Free and Open Source Software (FOSS) Usage in Military Computing

Track Topic: "C4ISR/C2 Architecture"

Richard Carbone, Robert Charpentier and David Demers
Defence Research and Development Canada - Valcartier

DRDC Valcartier
2459 Pie-XI Blvd N
Val-Bélair, QC
Canada G3J 1X5

Point of Contact: David Demers
Telephone / Fax: (T) 418-844-4000 x4601; (F) 418-844-4538
E-mail Address:David.Demers@drdc-rddc.gc.ca

Free and Open Source Software (FOSS) Usage in Military Computing

Richard Carbone, Robert Charpentier and David Demers
Defence Research and Development Canada – Valcartier

Abstract

Free and Open Source Software (FOSS) has been constantly growing in importance and expanding in many software architectures all over the world. This impressive growth has been supported by the numerous successes, the high-quality reputation of FOSS-based systems and significant cost savings. Moreover, many FOSS applications have achieved a level of maturity and of recognition that raises them to a position of superiority over their commercial equivalents. With the quality and the quantity that is now available, FOSS can be seen as a viable alternative to proprietary closed source software in many application domains.

This paper will present an overview of FOSS usage in the civilian world and will highlight the specifics of the military business case for more FOSS usage in C2IS architectures and for a better understanding of the potentially disruptive aspects of this technology. Key topics from a military perspective include: improved security through code access, easier long-term support, interoperability through implementation of open standards, increased confidence in open source as opposed to closed source software, and the need to understand "potentially offensive" software developed using a collaborative process via the Internet.

1 Introduction

In Fall 2003, Defence Research & Development Canada (DRDC) initiated a special study to determine the role of FOSS in the evolution of our information system architectures. Our prime objective was to provide DRDC project leaders with some guidelines in assessing the usefulness of FOSS in their specific project contexts. These recommendations were proposed with regard to the context of development and projects that are internal to DRDC (or through its contractors), as part of its mandate to explore advanced IT tools and resources, such as implemented in "proof-of-concept" demonstrations. Although FOSS may provide a promising approach in the deployment of actual operational resources within the Canadian Forces, such operational use and assessment of opportunity were beyond the terms of reference of the initial study.

The report went through three commentary and validation phases. The first phase was a revision by a panel of local experts at DRDC Valcartier, followed by reviews at DRDC Corporate Headquarters (second phase) and by Department of National Defence/Canadian Forces (DND/CF) personnel (third phase) and OGDs (Other Government Departments). The goal was to produce an objective evaluation of the utility of FOSS for DRDC (and also for usage within present and future CF/DND information systems) based on credible, factual sources.

One result from the revision of our report by OGDs has been to produce a "de-militarized" version of our study for adoption by all other Canadian departments in support of the new GoC (Government of Canada) position on FOSS. The official position of the GoC regarding FOSS usage was developed by Canada's Treasury Board Secretariat Chief Information Officer Branch and can be found at the following Web site: http://www.cio-dpi.gc.ca/fap-paf/oss-ll/position_e.asp. The policy is "FOSS-adoption neutral." To summarize, the policy requires that FOSS and Commercial-Off-The-Shelf (COTS) software solutions be evaluated on a level playing field. This is an evolution from the previous policy that favored information system solutions based on COTS components as the default (to break out of the 1970s and 1980s "custom software everywhere" phenomenon that was untenable on a long-term basis).

2 FOSS in Civilian Computing

2.1 What is FOSS?

During the past two decades, the software market has been dominated by COTS products that offer a myriad of functionalities at reasonable prices (e.g. Microsoft Windows and Oracle database management systems etc). However, the intrinsic limitations of COTS software have emerged over time (i.e. closed source code, expensive upgrades, lock-in effect, security weaknesses, etc.). This led to the development of a parallel "economy" based on free and open source software. After a slow beginning in the late 1990s, Free and Open Source Software (FOSS) has been constantly growing in importance and expanding in many software architectures all over the world. This impressive growth has been supported by the numerous successes, the high-quality reputation of FOSS-based systems and, of course, by the expectation of cost savings.

FOSS refers to software whose source code is made available for use and modification without the expensive license fees imposed by COTS software vendors. FOSS can be developed by volunteers or through development sponsored by large computer firms who want to include "commodity" software to give a competitive advantage to their hardware products, or whose core business is to provide support services for FOSS-based system implementations. Over the past ten years, FOSS development has grown. Thousands of FOSS projects are now carried out via Internet collaboration; hundreds of high-quality applications are available for use or modification at no (or little) cost and many FOSS products are now widely available that are considered to be as mature and secure as their COTS equivalents. Many very useful software products have been distributed using an open-source paradigm. Some of the better known ones are: the LaTeX text editor (and typesetting tool) used for scientific publications; the Linux operating system; Apache - a reliable and secure web server; and MySQL - a high-performance, reliable and full-featured database.

2.2 FOSS Legal Issues

Licenses attached to FOSS provide basic rights to the licensee and the user but they vary a great deal depending on the originator's preferences. Even if much freedom is given to the users of FOSS, the programmers can impose some constraints on the exploitation of the components that are being released. For example, a license can state that the released component can only be integrated into a pure FOSS system (i.e. purist approach), or alternatively, may be linked with some proprietary library (i.e. more practical approach). System architects must pay attention to the terms of the license when selecting a product in order to avoid legal pitfalls. License comparison can be found in many references [Wu01, Dri01, Bol03].

The most common license is the GNU General Public License (GPL), which is found to be used in about 65% of the available FOSS applications. The Gnu's Not UNIX (GNU) General Public License (GPL) has been adopted by the majority of programmers over the past 15 years and has been the reference model for many other license agreements. This licensing model demands that all complementary code development be integrated with open source software only and be published under a General Public License (GPL) compatible license. It is expected however that less restrictive license models (such as Mozilla and BSD) will be more popular in the future [Dri01], since hybrid proprietary/FOSS systems are more appropriate to most modern hybrid IT infrastructures.

Some complementary tactics were developed to prevent appropriation of FOSS by commercial firms. FOSS is intrinsically exposed to the risk of proprietary take-over by commercial vendors. Therefore, in addition to licensing, developers may legally incorporate and/or transfer their

property rights to a nonprofit corporation and/or trademark the brand and logos of their software product etc. [OMa03].

Patenting algorithms is intrinsically incompatible with FOSS development. According to Richard Stallman, the worst threat faced by FOSS comes from software patents, which can put algorithms and features off-limits to free software for up to twenty years. The Free Software Foundation (FSF) has a petition in circulation against software patents [CasWWW]. In [Whe03], the author indicates that Microsoft increased its patent rates by more than 30% in 2002, which may confirm that the threat is real.

2.3 Key advantages of FOSS

By its simplicity and efficiency, the FOSS development model has repeatedly demonstrated many benefits:

- Huge diversity of software [Spi04].
- High flexibility and scalability of software solutions through source code editing.
- High reliability and high security through source code review and validation [Dow00].
- One-order of magnitude faster release rate than equivalent COTS software.
- Rapid development of custom solutions to meet specific requirements through code reuse and extension.
- Lifetime extension of FOSS-based systems through source code upgrades [Cal03].
- High degree of compliance with open standards leading to more interoperability between information systems.
- Leaner and meaner systems compared to COTS equivalents that often suffer from "marketing feature bloating".

For more documentation on FOSS advantages, the following references are recommended [Eva03, CasWWW, Dow00].

FOSS has also evolved into a very efficient "development process." The strength of FOSS development is the ability to recruit and motivate communities of competent programmers to develop, debug, and optimize code on a volunteer basis. Coordination is assumed by a delegate leader who is responsible for the assessment of the various solutions offered by the programmers and for the integration of the best code into the next FOSS updates that are rapidly put on-line [UN03].

On the other hand, some criticisms can be found in scientific literature. When the DRDC report was being prepared (i.e. winter 2004), the following criticisms were found in the technical literature:

- Version control may be more complex with FOSS than COTS (evolving).
- System maintainability requires more local resources (debatable in the long-term).
- Higher technical skill needed from system administrators.
- May offer less integration within an application suite and less user-friendliness (evolving).

Some other criticisms can be found in various reports but in many instances the perspective is clearly biased. For example, the National Economic Research Associates (NERA) report claims that there is nothing wrong with closed source software since it allowed "a very fast growth of the software industry over the last few decades, providing ever more powerful, easy-to-use software to ever more users" [Bis04]. Even if NERA's impartiality can be questioned (since their study was sponsored by Microsoft), the report offers quite complete counter-arguments against FOSS adoption in government. This report is not unique! A recent article from Todd Bishop indicates that many studies on Linux performed by IDC, Giga and Meta Group were in fact sponsored by Microsoft. Critics question how independent the analyses were [Ver03]. Interestingly, the

January/February 2004 edition of IEEE Software includes a series of very positive articles on FOSS, which are preceded by a guest editor's introduction signed by Szyperski (Microsoft Research) and by Spinellis (Athens University of Economics and Business) [Spi04].

2.4 FOSS around the world and in the USA

Many countries around the world recognized the qualities of FOSS. For example, European Economic Community countries are actively adopting FOSS, in particular in the public sector. Latin American, African, Oceania and Asian countries are also moving toward FOSS to varying degrees. This site http://www.opensource.wa.gov.au/resourcesfldr/files/0408_ospolicies.pdf provides a summary listing of FOSS initiatives worldwide. The strategic rationale (at a national/governmental level) for migrating to FOSS is based on three main factors:

1. Direct cost savings – including reduction of the economic loss at the national (macro-economic) level caused by commercial software imports;
2. Access to source code to develop national IT expertise; and
3. "Controlling their IT destiny" – includes reduced dependence on external COTS suppliers for long-term systems and architectural evolution.

FOSS originated largely in the United States of America (USA) and remains a very strong movement there. In addition to the software developed by groups of volunteers, a substantial contribution to FOSS is made available by large firms who wish to experiment with a different business model based on collaborative development. Netscape has had one of the most famous success stories in adopting an openness strategy that is described in this very interesting article [Hec00]. IBM, Hewlett-Packard, Sun Microsystems, Novell and Silicon Graphics are just a few of the better-known IT leaders who host/contribute/sponsor/support a large number of open source projects [Fri03]. IBM made a formal commitment to speed Linux deployment in the banking industry [Jac02] and in government [Fis02].

Some USA government initiatives contribute to FOSS. Government sponsoring of FOSS is not common, although some examples are reported in [Bol03]. In Geomatics, the National Technology Alliance (NTA) has sponsored the impressive Open Source Prototype Research project, which had a significant impact on geospatial information organizations in the USA government [TYB02] including the Department of Defense (DoD). More recently, a mission-critical development with FOSS has been reported in IEEE Software [Nor04] and describes how FOSS has been used very efficiently in NASA JPL projects.

Adopting a strong FOSS policy may be problematic for the American government since the proprietary software industry strongly supports the USA economy. The software business is estimated at $70B (US) [UN03] annual revenues, and so it is not surprising to see a vigorous reaction from COTS editors against FOSS [Fry03]. For example, the reference [Eva03] is a Microsoft sponsored study that tries to counter the FOSS business case and a more formal article [Mun02] gives the official Microsoft perspective on FOSS.

2.5 FOSS in Canada

Canada appears to be behind the curve in adopting FOSS, especially in the public sector. At the present time, the use of FOSS in Canada is mostly in software development and in the back-office environment (i.e. servers and network management). It is expected that this trend will remain dominant for the next 1-3 years [Pat03].

The Government of Canada is now engaged in the process of adopting a balanced approach to ensure that governmental policies and guidelines do not bias one software business model over

another (FOSS vs. COTS vs. custom development). Some government departments were mandated to support the national policy on FOSS by specific actions like: to review federal procurement practices to ensure a level playing field; to develop a strategy for property rights, patent protection and technology transfer; to provide advice on licensing and other legal issues; etc. [GoC04].

2.6 FOSS Business Case in the Civilian World

Most forecasting firms predict continuous growth of FOSS. As mentioned earlier, the strategic rationale for migrating to FOSS is typically related to three main factors: 1) the expectation of direct cost savings, 2) the reduction of economic loss at the national level caused by commercial software imports and 3) the hope to better develop national IT expertise by means of access to source code (and development of original components) which is not generally possible with COTS packages.

FOSS offers the flexibility of building a specialized system in an accelerated development process that is at least as efficient as buying COTS components. For R&D projects, the use of FOSS can assure a rapid development (i.e. code reuse and modification) of a high quality code (i.e. well debugged), which would be very difficult to achieve through custom code developed from scratch. In some instances, FOSS-based development is the only reasonable alternative when COTS products are not available (e.g. High Performance Computing [Mor03]) and when custom development is too expensive for the available budget [Fry03]. FOSS helps in avoiding lock-in to proprietary IT products and services and in reducing our dependence on monopolistic technologies.

We must also highlight that very strong support also comes from the IT industry itself which offers very high-quality free software to support the commercialization of their hardware products. As mentioned by J. dal Molin in [Molin], software is often seen as a "commodity" that simply gives added value to products under strong competition. Other IT industry organizations have built their business model around providing support and integration services for FOSS-based information systems.

The underlining business model that collaborative software development promotes seems to be less understood at this time and is probably under-estimated. Early adopters (such as NASA's collaborative development of Mars exploration [Nor04]) have, however, greatly appreciated the generous contribution of highly skilled volunteers who enjoy contributing to prestigious development projects just for the fame or sake of it. Other success stories can be found in the education and health sectors where collaborative development was utilized [Mol03, Dum04]. Progressively, adoption of FOSS development methods will continue to impact on engineering practices with more fundamental and far-reaching consequences.

While very attractive in general, FOSS must be evaluated in the context of each project on a case-by-case basis in order to determine if the advantages outweigh the disadvantages in practice. Some preliminary guidelines for the Canadian Government are available in [Cha05]. An outline of this evaluation methodology follows:

- Define the application context – project objectives, client expectations, prioritize evaluation criteria etc.
- Identify candidates – COTS, FOSS.
- Compare the best 3 to 4 options side-by-side (see Appendix A for the (Draft Version) Canadian DND FOSS Product Summary and Checklist).
- If appropriate, perform an in-depth code analysis for FOSS candidates.

- Seek approval from client and local management.
- Document lessons learned

2.7 Important Ingredients for FOSS Success

In a recent annual report from the Software Engineering Institute [SEI02], several key factors for successful open source software adoption are described. Specifically, successful OSS is typically:

- A good working product,
- Led by committed leaders,
- Providing a general community service,
- Supported by developers who are also its users.

FOSS is often perceived as a return to more reliance on internal resources for system development and maintenance. For security enforcement, high-quality expertise is scarce and may often have to be developed to adequately cope with the increased responsibilities that FOSS-based systems will require. Access to source code can also be an advantage to an attacker who can try to develop more elaborate attacks on the open source code [Mun02]. Some authors are also concerned about potential infiltration into collaborative development projects by malicious developers who could install backdoors or other undesirable functionalities [Jon04]. At any rate, neither COTS nor custom software are immune to malicious or programming defects that result in information system vulnerabilities. There is no magic solution to reducing the risk of software vulnerabilities. Only a large community of trusted, skilled and motivated software developers and users will reduce an attacker's ability to develop a "zero-hour" exploitation for a FOSS-based information system.

These findings have important implications for the adoption of FOSS for military-specific (non dual use) applications. The potential benefits of FOSS usage will likely not be fully achieved for military extensions that are of interest to only a handful of (friendly) developers worldwide. One of the prerequisites for the creation of a robust FOSS component/application is a large community of stakeholders/users and developers. From a Canadian perspective, FOSS development for military specific applications within a collaborative framework such as a CanUS bilateral or AUSCANZUKUS project holds more promise that attempting to "go it alone."

3 OSS in Military Computing

The military business case is, for a large part, identical to the civilian one (i.e. cost savings, avoiding lock-in effects etc.). It also has some specificity that needs to be highlighted.

(A) Better security through code enhancement, code diversity and elimination of useless features

When software is created, it has a level of quality that depends directly on the programmer's competence, experience and professional methodology. To increase the reliability and security of code, it is essential to use some complementary mechanisms such as peer review, testing, quality audits, alpha and beta versioning etc. FOSS and proprietary software rely essentially on the same processes (probably at similar levels) during the main development period. However, after the first public release, FOSS offers the very significant advantage of permitting access to the source code. This encourages further peer reviews, testing, and quality audits by a much larger community of users/developers than what would be possible with proprietary code. For closed source software, flaws and code defects are often discovered by some subversive exploits which

can lead to some destabilization in large corporations that rely on such COTS packages (i.e. patch and repair). On the contrary, confidence in FOSS may be built faster and, potentially, to a higher degree than with a proprietary equivalent [His02, ITs02]. A myriad of statistics on software vulnerabilities are available in Chapter 6 of reference [Whe03] and they seem to confirm the general perception that open source software is often superior to proprietary code. Reference [Jiw02] gives a comparison of the vulnerabilities contained in Red Hat Linux (160) and Windows NT (1200) that appears to be more scientific but great care should be taken to avoid extrapolating this study beyond its original scope. In short, FOSS is not intrinsically more secure than COTS software but the openness of source code makes security enforcement more ubiquitous and less disruptive. The dilemma on security through obscurity vs. openness was the subject of a heated debate in the cryptographic community in the 1980's. The final decision was to make the cryptographic algorithms generally available so as to provide for security assessment and validation by the widest scientific community possible. Whitfield Diffie, the inventor of public key cryptography, and now chief security officer at Sun Microsystems, has repeatedly said that "openness is essential for trust" in software as it was for cryptographic protocols twenty years ago [Jud02, Dif03]. "Sunshine kills bacteria" [His02].

Other security advantages for FOSS include:
1. "Leaner and meaner" software systems than COTS equivalents that often suffer from feature bloating. Since they are smaller, open source systems are expected to provide fewer opportunities for exploits.
2. Source code can be enriched with assertions, complementary safety checks, etc.
3. Increased code diversity in the "software ecosystem" that could reduce the speed and the proliferation of cyber attacks.

In the USA for example, the DoD seems to be among the leading departments that are adopting FOSS. The business case for FOSS was first done in 2001 by MITRE [Ken01]. This "Strengths, Weaknesses, Opportunities, and Threats" analysis of Linux was probably the first coherent and rigorous study that illustrated the true potential of FOSS in a military context. A follow-on study published in 2002/03, recommended that concrete steps be taken to encourage FOSS adoption, such as the creation of a "Generally Recognized As Safe" FOSS list along with technical policies. The 251 documented examples of FOSS applications give an excellent idea of the importance of FOSS within DoD [Bol03]. MITRE analysts also concluded that FOSS, by virtue of the diversity of source code and the 'total' visibility of implementations, allows faster and more autonomous responses to cyber threats [Bol03].

Many other international initiatives can be found on the Internet that also aim at identifying high-quality free software [Cha05]. Appendix B gives a list of organizations that provide frameworks for FOSS certification and also components that went through some level of certification. It is expected that some pre-qualified software will be very popular in military architectures. Appendix C includes a preliminary list of FOSS offering some analysis capabilities to help improve security posture of networked information systems.

(B) Easier long-term support

Long-term system support issues are less difficult to address when the source code is open. Long-term maintenance for embedded systems (as an example) can be difficult when 2005 software updates are being performed for 1985 hardware. One can imagine that the same situation will occur for 2025 software updates on circa-2005 hardware within embedded systems. The ability to conserve the source code and compilers used to develop the target system is a great

aid. In addition, FOSS tends to be based on open standards; thus embedded hardware upgrades within larger systems of systems may be facilitated for FOSS based components.

(C) Maintaining Interoperability with Allied Nations Migrating to FOSS

FOSS implements open standards and specifications that are shared among developers during the design, coding and testing processes. This is generally recognized as a strategic advantage in enforcing interoperability policies between independently developed systems [OGC02].

For a country like Canada that performs more than 90% of its operations with allies, maintaining interoperability with other countries is mandatory - especially with the USA. Some nations have announced their intent to migrate their military systems to FOSS-based components (or that may be considering this in the near future). NATO is also defining some guidelines to allow FOSS usage in the future. [NATO04]

(D) Increased confidence in open source as opposed to closed source software

Many nations are moving to FOSS because of a lack of confidence in COTS software. For the same reason that the US and Canada would never procure their military information systems from Russia and China, most emerging nations do not trust "American" software products to build their C2IS. Their fear is reinforced by the fact that these commercial software products are distributed as executables where it is relatively easy to hide undesired functionalities like Easter eggs and trap doors. Since developing a competitive software industry is out of reach for these emerging nations, their only realistic option is to adopt FOSS components, harden them and integrate them into their C2IS.

It is therefore important to follow the usage of FOSS technology not only to maintain interoperability with FOSS-equipped allies (and friendly non-governmental organizations) but also to understand the capabilities of belligerents in OPS theaters.

(E) Understanding "potentially offensive" software

The FOSS collaborative process allows for the efficient creation of software; either defensive or offensive. A quick survey of the FOSS projects available on the Internet allowed identifying approximately 21 significant projects that are aimed at attacking information systems. The tools developed within the first 6 projects described within Appendix D may be very serious threats to military information systems deployed in a theater.

Like any other technology that is being used in a conflict, understanding belligerents' capabilities is a key to success. Given that the tools derived from FOSS may be used, and that sometimes the results of the use of hacking/cracking tools can have unforeseen side-effects, it is imperative to maintain, at a minimum, a technology watch of activities over the potential for FOSS-based tools in a military context.

4 Trends and Issues with FOSS in Military Computing

FOSS is not a panacea, but it does offer concrete opportunities for cost saving, technology insertion and flexibility. Among FOSS benefits, an augmented compliance with open standards will contribute directly to military system interoperability; higher security will be achievable via source code auditing. FOSS-based systems will continue to expand to a position of superiority over their commercial equivalents for many applications. To take advantage of these

developments, military organisations should consider more diverse use of software (custom code vs. FOSS vs. COTS).

Given that recent past, present and future military interventions/operations have been at varying conflict intensity levels, it is reasonable to assume that we will encounter the use of FOSS by our allies and by belligerent organizations. Thus, it will be important to follow the usage and capability trends of FOSS technologies in order to interoperate with FOSS-equipped allies (and friendly non-governmental organizations) and understand the capabilities of belligerents.

5 Acknowledgements

The authors wish to thank Second Lieutenant Stéphane Fortin for his help in preparing the appendices.

6 References

[Bis04] Bishop, Todd (2004). Studies on Linux help their patron: Microsoft. *Seattle Post-Intelligencer*.

[Bol03] Bollinger, Terry (2003). Use of Free and Open-Source Software (FOSS) in the U.S. Department of Defense. (Technical Report MP02W0000101 v1.2.04). MITRE.

[Cal03] Calvin, James B. and Rodgers, Steven L. (2003). The Case for Open Source Tools is Compelling. *COTS Journal*, pp. 25–29.

[CasWWW] Casamento, Gregory. Petition Against Software Patents. Web Page, http://www.petitiononline.com/pasp01/petition.html.

[Cha04] Charpentier, Robert and Carbone, Richard (2004). Free and Open Source Software-Overview and Preliminary Guidelines for the Government of Canada, External Client Report ECR 2004-232, Defence R&D Canada, September 2004.

[Dif03] Diffie, Whitfield (2003). Perspective: Decrypting the Secret to Strong Security. *news.com*.

[Dow00] Dowling, Ted (2000). Software COTS Components Problems, And Solutions?. In *RTO SCI Symposium on "Strategies to Mitigate Obsolescence in Defense Systems Using Commercial Component"*, pp. 28–1 to 28–8.

[Dri01] Driver, M. (2001). The Future of Open-Source Software. (Technical Report SPA-13-7536). Gartner.

[Dum04] Dumais, Michel (2004). Le logiciel libre en éducation: le projet MILLE. *Accération du CRIM*, **4**, 26–27.

[Eva03] Evans, David S. and Reddy, Bernard (2003). Government Preferences for Promoting Open-Sources Software: A Solution in search for a problem. (Technical Report 9 Mich. Telecomm. Tech. L. Rev. 313 (2003)). National Economics Research Associates.

[Feller et al. 02] Feller, J. & Fitzgerald, B. *Understanding Open Source Software Development* (ISBN: 0201734966). Boston, MA: Addison-Wesley Professional, UK, March 2002.

[Fis02] Fisher, Mary Ann (2002). Linux in Government White Paper. Paper. IBM.

[Fri03] Fricke, Pierre (2003). Linux Strategies and Solutions 2003: Linux Server Suppliers Contend for Leadership. Technical Report. D.H. Brown Associates, Inc.

[Fry03] Frye, Emily (2003). Open-Source Software, Proprietary Software: Implications for National and Economic Security. Technical Report. THE CIP REPORT.

[GoC04] Branch, Chief Information Officer (2004). GoC Proposed Position on Open Source Software and Next Steps. Presentation Power Point. Treasury Board Secretariat of Canada.

[Hec00] Hecker, Frank (2000). Setting up Shop: The Business of Open-Source Software. Paper. Netscape.
[His02] Hissam, S.A., Plakosh, D., and Weinstock, C. (2002). Trust and Vulnerability in Open Source Software. (Technical Report 20020208). IEE Proceeding online.
[ITs02] ITsecurity (2002). The Strengths and Weaknesses of Open Source Software - And Its Role in the Security Model. Paper. ITsecurity.com:.
[Jac02] Jacob, Bart, Janson, David, Mark, Oliver, and Marras, Fabio L (2002). Linux and Branch Banking. (Technical Report SG24-6909-00). ibm.com/redbooks.
[Jiw02] Jiwnani, K. and Zelkowitz, M. (2002). Maintaining Software with a Security Perspective. (Technical Report IEEE). International Conference on Software Maintenance (ICSM'02).
[Jon04] Jones, A. Russell (2004). Open Source Is Fertile Ground for Foul Play. DevX.com.
[Jud02] Judge, Peter (2002). Diffie defends open-source security. *ZDNet, UK*.
[Ken01] Kenwood, Carolyn A. (2001). A Business Case Study of Open Source Software. (Technical Report MP 01B0000048). MITRE.
[Mol03] Molin, Joseph dal (2003) Open Source Software in Canada – A Collaborative Fact Finding Study. Technical Report. e-cology Corporation.
[Mor03] Moran, Patrick J. (2003). Developing An Open Source Option for NASA Software. (Technical Report NAS-03-009). NASA Ames Research Center.
[Mun02] Mundie, Craig (2002). Security: Source Access and the Software Ecosystem. Technical Report. Microsoft Corporation.
[Nor04] Norris, Jeffrey S. (2004). Mission-Critical Development with Open Source Software: Lessons Learned. (Technical Report vol 21 no1). IEEE Software.
[NATO04] Infosec Sub-Committee, Security Implications of Using Open Source Software in NATO, ac/322(SC/4)N(2004)0006 (INV) v 0.9 January 2004
[OGC02] Office of Government Commerce (2002). Open Source Software – Guidance on Implementing UK Government Policy. Technical Report. Cabinet Office Minister of State.
[OMa03] O'Mahony, Siobham (2003). Guarding the Commons: How Community Managed Software Projects Project Their Works. Paper. Harvard University Graduate School of Business Administration.
[Pat03] Patrick, Ryan B. (2003). Linux deepening its Canadian foothold. *Computerworld*.
[PITAC 00] President's Information Technology Advisory Committee (PITAC), Panel on Open Source Software for High End Computing: co-chairs Smarr, L. & Graham, S. "Developing Open Source Software to Advance High End Computing" [online]. <http://www.ccic.gov /pubs/pitac/pres-oss-11sep00.pdf> (September 11, 2000).
[Raymond 99] Raymond, E. *The Cathedral & the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary* (ISBN: 1565927249). Cambridge, MA: O'Reilly & Associates, October 1999.
[SEI02] SEI Independent Research and Development Projects, Technical Report, CMU/SEI-2002-TR-023, ESC-TR-2002-023, Carnegie Mellon Software Engineering Institute, October 2002.
[Spi04] Spinellis, Diomidis and Szyperski, Clemens (2004). How Is Open Source Affecting Software Development?. Paper. IEEE SOFTWARE.
[TYB02] TYBRIN (2002). Open Source Prototype Research – Open Source Process. Technical Report. NTA – NCAT.
[UN03] United Nation Conference on Trade and Development Secretariat (2003). Free open-source software: Implications for ICT policy and development, Ch. 4. United Nations.
[Ver03] VeriTest (2003). Microsoft Windows Small Business Server 2003 vs. Red Hat Enterprise Linux ES 2.1 Deployment. Test Report. VeriTest.
[Whe03] Wheeler, David A. (2003). Why Open Source Software / Free Software (OSS/FS)? – Look at the Numbers!. Paper. Personal Web Page.

[Wu01] Wu, Ming-Wei and Lin, Ying-Dar (2001). Open Source Software Development: An Overview. (Technical Report IEEE 0018-9162/01). National Chiao Tung University, Taiwan.

Appendix A

Canadian Department of National Defence

FOSS Product Summary and Checklist

(Draft Version)

DRAFT

FOSS Product Summary and Checklist

Are there any guidelines for filling out this questionnaire? Should this be attached to any and all RFCs requesting this type of software? If so, some guidelines or an intro should be included here.

Part A – Project Objectives & Context

Project and RFC Initiator Contact Information		
Initiator Name & Designation:		
Telephone/Contact Number:		
Project Partners and Collaborators:		
Name: Responsibilities:		
Name: Responsibilities:		
Name: Responsibilities:		
ISSO Contact Information (Information Systems Security Officer **OR** Designate)		
ISSO Name & Designation:		
Telephone/Contact Number:		
Project Objectives and Context		
Project Title:		
Project Overview:		
Security Level: Note: Combining unclassified data releases could lead to a security problem.		
Execution Environment:	Prototyping:	
	Deployed Version:	
Development Environment: If additional code needs to be developed		
Compliance with Standards:		
Compatibility with legacy data/information formats: Ability to interface with existing informatics' systems		
Recommended Code Review: Yes/No Level required		
Intellectual Property: Protection Issues		
CF Client Preferences:		

PART B – FOSS & COTS SOFTWARE DESCRIPTION

Comparison Criteria for FOSS vs. COTS Software	
Criteria	***Comments***
Functionality Do the features of the product meet user and business requirements?	
Cost List any costs associated with the implementation of this product: licensing, installation, maintenance, training *etc*.	
Required Support & Maintenance Are the proper means in place in order to adequately support users and the evolution of the tool? i.e. training, maintenance, upgrades etc.	
Reliability Can the product/tool adequately support and deliver the necessary functions to its users? (REF: Wheeler, GRAM, GRAS)	
Quality Efficiency, reliability and productivity of the tool and its features.	
Ease of Migration If migrating from a previous product or tool, what is the impact to systems and users?	
Performance & Scalability Has user acceptance testing (UAT) been conducted (using test case scenarios based on every day business transactions/use) to ensure needs are met?	
Flexibility & Scalability Is the software customizable in order to fulfill current organizational needs? Is the software/tool scalable and can it evolve with your needs?	
User Friendliness Is the software's functionality and user-interface (GUI) intuitive and easy to use?	
Developer Usability How easy is it for developers/programmers to customize, control and maintain the software? Is the application programmer interface (API) managed via a command-line interface or GUI that is user friendly?	
Legal & Licensing Issues How does the licensing for this OSS/FS product work? Does the licensing vary? i.e. is it different for users and programmers? How easy is it to acquire them?	

Comments

PART C – FOSS & COTS PRODUCT COMPARISON MATRIX

Criteria	Importance Ranking*	Product 1		Product 2		Product 3	
		Eval	Comments	Eval	Comments	Eval	Comments
Functionality							
Cost							
Support & Maintenance							
Reliability							
Quality							
Ease of migration							
Performance & Scalability							
Flexibility & Scalability							
User friendliness							
Developer Usability							
Legal & Licensing Issues							

***Importance Ranking**
A – Essential
B – Important
C – Nice to have

Appendix B: FOSS Certification and Trusted Sources

(in alphabetical order)

1. Common Criteria

http://www.commoncriteriaportal.org/public/consumer/index.php?menu=7

Countries :
Australia, New Zealand, Canada, France, Germany, Japan, Netherlands, United Kingdom, USA
Product(s) Example(s):
Security certifications from EAL1 up to EAL7
Description:
The Common Criteria represents the outcome of efforts to develop criteria for evaluation of IT security that are widely useful within the international community. It is an alignment and development of a number of source criteria: the existing European, US and Canadian criteria (ITSEC, TCSEC and CTCPEC respectively). The Common Criteria resolves the conceptual and technical differences between the source criteria. It is a contribution to the development of an international standard, and opens the way to worldwide mutual recognition of evaluation results.

2. FSG (Free Standards Group)

http://www.freestandards.org/

Country:
United States
Product(s) Example(s):
LSB (Linux Standard Base), Openl18N (The Free standards Group Open Internationalization Initiative)
Description:
The Free Standards Group is an independent, nonprofit organization dedicated to accelerating the use of free and open source software by developing and promoting standards. Their main projects, LSB and Openl18N, seek standardization so that the different Linux (Open Source platforms) can seamlessly interoperate, thus minimizing risks for businesses and agencies wanting to use aforementioned systems.

3. MandrakeSoft

http://www.mandrakelinux.com/en-us/

Country:
France
Product(s) Example(s):
MandrakeLinux
Description:
MandrakeSoft is one of the major Linux distributors and contributors, on par with Red Hat and SUSE. MandrakeSoft offers server and user side versions of its Operating System as well as customized versions for specific purposes such as Firewalls. Mandrakelinux has always been known as one of the most (if not the most) innovative Linux distributions, including all the latest technologies from the Open Source community. Mandrakelinux has partnered with Conectiva, Progeny and Turbolinux to create a common implementation of LSB 2.0, an Initiative under the name of LCC (Linux Core Consortium). This initiative is backed by Linux supporters such as Computer Associates, HP, Novell, Red Hat, Sun, OSDL and FSG.

4. Mozilla

http://www.mozilla.org/
http://www.mozilla.org/foundation/

Country:
United States
Product(s) Example(s):
Mozilla, Firefox, Thunderbird, Camino
Description;
Mozilla is a community sponsored by the Mozilla Foundation, once part of the AOL Netscape division. The Mozilla Foundation exists to provide organizational, legal, and financial support for the Mozilla open-source software project. Firefox is the most popular and fastest growing product. Their solutions are not limited to the Microsoft Windows platform, but extend to Linux, Solaris, Mac OS X, and more through the source code. Their web browser, using the Gecko engine, is the same as what is driving Netscape. It is recognized as a secure web browser whose usage has been accepted by government and military departments.

5. Novell

http://www.novell.com/

Country:
United States
Product(s) Example(s):
SUSE Linux Enterprise Server 9
Description :
Novell is a software information, applications, processes and systems integration company who has acquired SUSE recently, positioning itself as a Linux market leader. SUSE Linux Enterprise Server has been certified Common Criteria EAL4, making it the first Linux system to receive such a high security certification.[1] Novell is well known for its NetWare suite being widely used in different enterprise and governments agencies.

6. Red Hat

http://www.redhat.com/software/rhel/
http://www.redhat.com/solutions/industries/government/commoncriteria/

Country:
United States
Product(s) Example(s):
Red Hat Enterprise Linux
Description:
Red Hat is known as one of the leading forces behind Linux. Its business now revolves around its enterprise software which is certified EAL3. Red Hat is often the de facto choice in North America for Linux servers/workstations.

[1] Robert W. Smith, **First security evaluation in compliance with Common Criteria EAL4+ for Suse Linux**, heise online news, 16 February 2005, http://www.heise.de/english/newsticker/news/56451 .

7. SELinux

http://www.nsa.gov/selinux/
http://www.nsa.gov/selinux/info/contrib.cfm
http://fedora.redhat.com/projects/selinux/

Country:
United States
Product(s) Example(s):
SELinux, Fedora Core 3, Red Hat Enterprise Linux 4, SUSE, Debian, Gentoo
Description :
SELinux (Security-enhanced Linux) *is a research prototype of the Linux® kernel and a number of utilities with enhanced security functionality designed simply to demonstrate the value of mandatory access controls to the Linux community and how such controls could be added to Linux. The Security-enhanced Linux kernel contains new architectural components originally developed to improve the security of the Flask operating system. These architectural components provide general support for the enforcement of many kinds of mandatory access control policies, including those based on the concepts of Type Enforcement®, Role-based Access Control, and Multi-level Security*.[2] SELinux has started being implemented and/or supported by some of the major Linux distributions such as Fedora Core 3, the new entity previously known as Red Hat Linux, or SUSE, Debian and Gentoo Linux.

8. Sun Microsystems

http://www.sun.com/software/linux/community/

Country:
United States
Product(s) Example(s):
OpenOffice, NetBeans, Project Looking Glass, GNOME, Mozilla, Apache, Project JXTA
Description:
Sun is one of the large contributors to the Open Source community. Known for its UNIX products and Operating System, Sun has also made its entry in the Linux world with the Java Desktop System, based on SUSE 8.0, but now part of Solaris 10. Some of the most important and well-known contributions Sun has made to the Open Source community are OpenOffice, Project Looking Glass, Mozilla, GNOME, NetBeans, Apache, and Grid Engine Project. Another important contribution would be SunSource.net, a site devoted to Sun's involvement in free and open Source projects.

[2] National Security Agency, http://www.nsa.gov/selinux/index.cfm, March 2005

Appendix C: Potentially Defensive FOSS

Top Tools

(in alphabetical order)

1. AntiSniff

http://packetstormsecurity.nl/sniffers/antisniff/

A tool designed to determine if a given system on the local network is running in promiscuous mode.

2. Arpwatch

http://www-nrg.ee.lbl.gov/

A network monitoring tool capable of mapping MAC addresses to IP addresses and is able to determine when IP or MAC addresses change on the network, as well as detect new network systems and detect ARP flood attacks.

3. EtherApe

http://etherape.sourceforge.net/

A GNOME-based graphical network traffic monitoring tool that shows network flows and communicating systems in a meaningful and useful way to the user.

4. Firewall Tester

http://www.infis.univ.trieste.it/~lcars/ftester/

A test suite for testing your network's firewalls and IDS/NIDS systems where the tool consists of an injector and a sniffer. While the injector injects traffic onto the network, the sniffer listens on the other side of the firewall or IDS/NIDS to determine if the injected packets can get through.

5. ISIC

http://www.packetfactory.net/projects/ISIC/

A highly configurable pseudo-random generating TCP/IP stack-testing suite that can be used to test the TCP/IP stacks of systems on the network.

6. Netfiler/Ipchains/Iptables

http://www.netfilter.org/

Quite possibly the world's most widely used computer firewall software that comes bundled with Linux and can transform very modest computers into powerful packet-filtering firewalls that can support NAT, bandwidth monitoring, and QoS.

7. Ntop

http://www.ntop.org

It is a Web-based graphical interactive network traffic management/measurement/monitoring tool that can also be used as a lightweight NIDS.

8. OpenBSD

http://www.openBSD.org

The most stable of the open source operating systems and arguably more stable than many commercial operating systems because of the ongoing static source code analysis and audit of the OS source code to find and plug all major security holes and gaps.

9. Pchar

http://www-nrg.ee.lbl.gov/

A rewrite of Patchar that is capable of performing network latency and bandwidth tests between the local system performing the test and a remote system that can be either across the LAN or across the Internet.

10. PortSentry

http://sourceforge.net/projects/sentrytools/

A real-time port scanning detection system that can block remote scans and has shown itself to be very effective against remote scanners and is multiplatform.

11. SELinux

http://www.nsa.gov/selinux/index.html

A NSA prototype, it is a hardened version of the Linux OS kernel that implements some of the features found in trusted operating systems such as MAC, ACL's, including least privilege and a rudimentary RBAC, making it ideal for security-conscious environments.

12. ScanLogD

http://www.openwall.com/scanlogd/

A TCP port scanning tool that is triggered when a given number of port scans occur within delta time where the tool does not actually stop the scan but instead logs it to the local Syslog facility in order to keep an audit trail of remote scans.

13. TCP Wrappers

ftp://ftp.porcupine.org/pub/security/index.html

A simple yet effective and powerful TCP/IP Inetd wrapping mechanism for network-based services/applications that verifies if a remote system has the right to use a TCP/UDP-based network service, port, or application, and adds virtually no overhead.

14. The Coroner's Toolkit

http://www.fish.com/tct/

A series of UNIX tools and scripts designed for the capture of processes, memory, and system states to be gathered from a running system before turning it off for post-mortem analysis by a qualified team of analysts/security personnel.

Appendix D: Potentially Offensive FOSS

Top Six Threats

(in alphabetical order)

1. DSniff

http://monkey.org/~dugsong/dsniff/

DSniff is a collection of powerful networking tools that were originally designed for network auditing. However, the tools can just as easily be used for malicious purposes. The DSniff, Filesnarf, Mailsnarf, Msgsnarf, Urlsnarf, and Webspy tools are able to passively monitor network communications in order to intercept a variety of different types of information such as passwords, e-mails, file sharing data, and others. There are other tools included such as Arpspoof, Dnsspoof, and Macof. These are tools that work on the 2nd layer of the TCP/IP stack. These tools enable a user to be able to hijack communications between two systems, and it is possible to use these tools to create relays. A relay is a system which acts as and appears to be the valid remote host in a connection, but in fact positions itself as a "proxy" go-between for communications with a valid remote system. Relays are virtually invisible to the average user. And finally, the tools Sshmitm and Webmitm are tools to be used to implement MITM attacks against *SSH* and HTTPS/SSL connections. DSniff is a very dangerous tool in the wrong hands.

2. Ettercap

http://ettercap.sourceforge.net/

Originally designed as a network sniffing tool for switched environments, it achieves its goal by spoofing and poisoning the network ARP caches, and can even flood them and overload them. This enables the tool to be able to sniff onto any system it wishes. Because of the ARP protocol, it is possible to perform man-in-the-middle (MITM) attacks which spoof/poison the ARP cache in order to defeat the complexities of a switched network. By performing MITM attacks, it can bypass the security found by network switching and sniff systems as if they were connected to each other. Ettercap also supports dissection of ciphered and normal network streams and it also has OS fingerprinting capabilities. Through its MITM attacks it can intercept username and password information from a variety of different network TCP/IP protocols, services, and applications. This tool should be considered extremely dangerous because it is very devious. Ettercap is also capable of smashing the TCP/IP stack of network switches and can cause all network traffic to be treated as if it were in a hubbed environment.

3. Kismet

http://www.kismetwireless.net/

This tool should be considered the de facto open source standard WiFi wardialling tool. It is a WiFi network detector, sniffer, and IDS system that is completely passive and can determine WiFi access points and collect WiFi network packets. It can even infer and find hidden network devices as well as standard named networks, and capture GPS data from various GPS-enabled network devices for future network mapping. More specifically, Kismet is an 802.11 network-based tool and works at Layer 2 of the TCP/IP protocol stack. The tool can also help to de-cloak features of the WiFi networks it encounters. Kismet is quite possibly the favorite WiFi detection tool of war drivers. Furthermore, due to the architecture of WiFi networks, there is very little which can remain hidden from Kismet, and when combined with other tools such as "John the Ripper" or DSniff or Ettercap, large amounts of information are put at the fingertips of potential attackers.

4. Nemesis

http://www.packetfactory.net/projects/nemesis/

Nemesis is a command-line tool designed to be a packet crafting and injection utility for the UNIX operating system. It is an easy-to-use "Shake'n'Bake" product because almost anyone with a basic knowledge of networking and TCP/IP can be up and running within a matter of a few minutes and attacking various remote systems and networks. In essence, it is a multi-protocol injection tool that is able to craft specially designed data packets that can be used for a variety of uses and reasons, ranging from analyzing your own network security to bringing down remote systems and networks. The tool allows you to build and send data packets that are carrying arbitrary payloads (from your own data files) to be used to test your network's resilience, as well as its ability to detect these malicious packets, or you could even use the tool to attack and destroy someone else's network.

5. Nessus

http://www.nessus.org/index.php

Nessus is the premier open source vulnerability security scanner. You should consider it the de facto standard for what a security/vulnerability scanner can and should do. Nessus can: 1) scan remote systems for network services, daemons, and applications; 2) determine the vulnerabilities of remote hosts, and 3) provide a facility to write and create your own vulnerability plugins. It works by determining the services and ports that are available on a remote host and then it will attempt to perform its security tests against these remote services. These attacks will help you to determine if the services, applications, and daemons you are running are susceptible to the various vulnerabilities in its plugin database. It currently supports more than 7,000 security plugins and should be considered extremely dangerous because of its accuracy, which lets potential remote attackers learn about the weaknesses and vulnerabilities of your systems.

6. Nmap

http://www.insecure.org

It is probably the world's most advanced port scanner. Nmap cannot only accurately determine the network services and applications running on a remote system but it can also very precisely fingerprint remote operating systems. What sets Nmap apart from many other similar tools is its ability to run in stealth mode and its speed. Nmap, when compared to other similar tools is very fast. When performing stealth scans it can either bounce scans off another system so that another system appears to be the one performing the scans, and it can run its scans at various speeds so as not to trigger any network traffic measurement tools running on the network. It is constantly being updated by individuals all around the world and thus it is always able to continue detecting the most recent operating systems because users push their new findings towards the project for inclusion into future releases. Furthermore, Nmap is also able to determine the ACK and SYN sequencing of your system's TCP/IP stack with which a competent attacker could understand and be able to determine how hard it would be to hijack your connection.

Other Potentially Offensive tools

7. Arping

http://www.habets.pp.se/synscan/programs.php?prog=arping

A powerful tool that exploits the usefulness of the ARP protocol and is able to determine if a given system is a network even if it the network is non-routable.

8. Attack Toolkit

http://www.computec.ch/projekte/atk/

A Win32 attack/penetration testing/auditing system that resembles Nessus and MetaSploit but that only has a portion of the security checks of Nessus. When compared to other open source Win32 vulnerability scanners, it is to be considered far superior.

9. Cheops/Cheops-NG

http://www.marko.net/cheops/

The "Swiss army knife" for UNIX networks, it is a powerful network-mapping tool that allows the user to map the local and remote networks in a manner similar to Win32 Network Neighbourhood and to scan all the devices on a network for their various properties.

10. Ethereal

http://www.ethereal.com/

The most powerful and configurable network packet sniffer available in both the commercial and open source worlds, it is without a doubt the de facto standard for protocol analyzers/sniffers and currently supports the largest number of protocols compared to any other tool currently available.

11. Firewalk

http://www.packetfactory.net/projects/firewalk/

An advanced Traceroute program that can traverse through firewalls to determine the full path to a given destination across both local and remote networks, including the Internet, and can also be used to map out local and remote network devices, routes, and paths.

12. Fragroute

http://www.monkey.org/~dugsong/fragroute/

A proof of concept tool designed to demonstrate that it is possible to insert packets into the network that can go undetected by IDS/NIDS systems because of the tool's use of packet fragmentation.

13. Hping2

http://www.hping.org/

A TCP/IP packet assembler/analyzer that can send files covertly through firewalls and is generally used to test out firewalls and IDS/NIDS systems, although it can also be used as a port scanner with OS detection, MTU discovery, and network stack auditing.

14. Modem Login Hacker

http://www.thc.org/releases.php

A wardialling tool that is compromised of scripts built for use with the UNIX modem tool Minicom that can dial a given phone number and try out a series of user-defined (from file flat) usernames and passwords against a remote system.

15. NBTScan

http://www.inetcat.org/software/nbtscan.html

A network scanning tool capable of finding NetBIOS/Win32 or other compatible systems such as Samba on a network. It reports information on the logged in user, MAC and IP information, and the domain or workgroup the remote system belongs to.

16. Netcat

http://www.securityfocus.com/tools/137

A simple tool capable of reading and writing directly across the network by using raw sockets. It can listen to or send a data stream (ideal for scripting), and it can also be used to determine if a given port/service/application is available on a remote system.

17. Strobe

http://www.rpmfind.net/linux/rpm2html/search.php?query=strobe

A rudimentary port scanner that works by connecting to all of the ports of a remote host that is neither fast nor efficient. It is, however, easy to use, and thus very commonly found.

18. Tcptraceroute

http://michael.toren.net/code/tcptraceroute/

A TCP-based rewrite of the UNIX Traceroute tool (but without the limitations of ICMP). It uses TCP packets encoded with ICMP query data information and thus generally passes through firewalls unhindered revealing more information than Traceroute.

19. THC-Amap

http://www.thc.org/releases.php

A port scanner that, instead of fingerprinting remote operating systems, attempts to determine the available network services and applications on a remote system from the data gathered during the establishment of a connection to a remote port.

20. THC-Scan

http://www.thc.org/releases.php

The standard for modern wardiallers that can dial a series of phone numbers either randomly or in series to detect remote systems such as modems, fax, voice-mail-box's, PBX, carrier, and tone, as well as find the data, stop, and parity bits of remote modems.

21. Xprobe2

http://www.sys-security.com/html/projects/X.html

A tool that uses fuzzy logic and a signature database to fingerprint remote operating systems with a very high level of accuracy.