JOURNAL OF AEROSPACE COMPUTING, INFORMATION, AND COMMUNICATION
Vol. 4, September 2007

Open Source Software: Free is not Exactly Inexpensive

Dr. Ben A. Calloni, P.E.*, John F. McGowan, J.D.† and Randall Stanley‡
Lockheed Martin Aeronautics Company, Fort Worth, Texas, 76101, USA

DOI: 10.2514/1.23080

Open Source Software (OSS) refers to software that is freely transferable to other users without charge. It is seen by many in the software industry as a mechanism to promote cost effective re-use and indeed it can reduce development costs and schedules. However, the use of OSS carries the potential of copyright infringement, which typically is not mitigated by supplier indemnities. Too many software developers rarely think about licensing issues when downloading a "good piece of free code" or a "free application". Further if one is using OSS for Mission / Safety Critical or Information Assurance, there are additional product certification processes that can add enormous cost to the development effort. It is not the authors' intention to provide a balance trade study or pro-con discussion of OSS vs. proprietary software but to raise the key points based on copyright and licensing compliance, and safety / security assurance issues that potential users must consider in using OSS. These problems are not usually at issue in purchasing proprietary software since appropriately authorized IT and legal personnel are involved in these purchases which include the licensing rights and Terms and Conditions of use from the vendor. All these risks have to be assessed and appreciated before one can say "it is free and it is cost effective!"

I. Open Source Software (OSS), Cost Panacea or Problem

OPEN Source Software refers to software that is freely transferable to other users without charge, but often there are conditions placed on the user of the software, either in how the software can be used or what must be done if the software is modified or incorporated into software which was developed by the user. Each OSS license agreement carries its own restrictions. Some, such as Apache, only require notices be posted. Others, such as GNU, have lengthy and onerous restrictions. For instance, any distribution of GNU with either OSS code directly used or statically linked to the company generated source code will require distribution of the company source code also as OSS and the posting of lengthy disclaimers and notices. Linux is licensed under GNU and other licenses. Whether the true owner of the software has granted the appropriate license is often unclear. Users must ensure that any use of OSS does not place it at risk of infringement and inclusion in its products does not compromise either the copyright integrity of the user's ownership nor require users to disclose valuable trade secrets in return for using the OSS software. At present, OSS is neither endorsed nor discouraged by the U.S. Government across the spectrum of project contracts. Some agencies actively promote its use while others are silent. This unresolved status makes program, project, and contractor decisions regarding OSS a point to consider within the context of a particular customer's directions. This

Received 17 July 2006; revision received 20 July 2007; accepted for publication 20 July 2007. Copyright 2007 by Dr. Ben A. Calloni, P.E., John F. McGowan, J.D., and Randall Stanley. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission. Copies of this paper may be made for personal or internal use, on condition that the copier pay the $10.00 per-copy fee to the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923; include the code 1542-9423/04 $10.00 in correspondence with the CCC.
* Lockheed Martin Fellow, Software Security, PO Box 748/MZ 8604, Member.
† Associate General Counsel for Intellectual Property, PO Box 748/MZ 1237.
‡ Principle Software Engineer, Software Methods and Architecture, PO Box 748/MZ 8604.

paper discusses the issues and costs associated with the use of OSS in general and specifically within the safety critical realms of avionics.

II. The Legal Issues

Any usage of software carries with it the requirement to be knowledgeable of the copyright notifications with the product and to be in compliance with the licensing rights to all software. This legal requirement exists regardless of whether the product is proprietary or OSS.

A. Copyright Law

Any discussion of Open Source Software must start with a basic understanding of copyright law. A copyright protects an original work of authorship by giving its owner the right to exclude others from reproducing, creating derivative works, distributing, performing or displaying the copyrighted work. This right exists from the moment the work is created and is fixed on some type of tangible media, such as paper, film, tape, computer disk, hard drive etc. Many different types of works of authorship may be protected by copyright such as literary works; musical works; dramatic works; pantomimes and choreographic works; pictorial, graphic, and sculptural works; motion pictures; sound recordings and, of course, computer programs. No further action is required to obtain a copyright. Registration with the Library of Congress is not required for the ownership rights to be established. Registration does give the owner certain rights in enforcing the copyright, which are not relevant to this discussion. The term of a copyright for works created after Jan. 1, 1978, (effective date of the current Copyright Act) is the life of the author plus 70 years. For works made by employees of a corporation the duration of copyright is 95 years from publication or 120 years from creation, whichever is shorter. Corporations usually require employees, as a condition of employment, to assign their copyright to the corporation for any works created related to their work. Often this assignment is for works created anywhere and anytime, if related to the corporation's business. Thus, a software engineer who writes code for his favorite OSS project may in fact not own the copyright to the software s/he creates and thus does not have the right to put the software under the OSS license.

There are penalties for infringement of copyright. Each open source file is a separate copyrighted item. The minimum statutory damages awarded, if the user was not on notice the work was copyrighted, is $750. With a copyright notice, the penalties could be as much as $30,000. Willful infringement, that is to say knowingly using work without permission, could cost the company $150,000 per occurrence. Open source products contain thousands of files and if 1% of a 10,000 file product contained infringing material, this could mean statutory damages in the range of $37,500 to $7,500,000.

B. Using Copyrighted Works

In order for someone to use the copyrighted work of another, they must have permission of the copyright owner to do so. This permission is called a license. The copyright owner has the right to prevent others from copying, distributing, publicly performing or making derivative works (more on this later) of the copyrighted work. The copyright owner can grant all of these rights or only some of them. Licenses can be drafted to grant rights in an almost infinite number of ways. For example, a software company may license a computer company to make copies of its operating system for installation in computers shipped from the factory, but not the right to make copies to be sold separately in stores. When you buy a movie DVD (digital video disc), you get a license to show the movie in your home for family and friends but not to publicly show the movie in a theater, which is a separate license right. In theory a license could be written so as to allow the user to run the software on one computer, one day a month, for five minutes, or any number of other limitations. There is no such thing as a "standard" license agreement. In one example, the OSS was licensed freely, provided the user had purchased a copy of the author's book as well. Each license must be read carefully to understand exactly what rights the copyright owner is granting the user. Licenses do not have to be printed and signed by the parties and are enforceable if you click the "I agree" button on the screen, open the shrink wrap and use the software or download and use the software. The license terms apply whether you read them or not.

The copyright owner has the right to prevent others from making a "derivative work", but what is meant by a "derivative work"? A derivative work is a copyrightable work that is based on another copyrighted work. An example would be a motion picture based on a novel. While the motion picture studio can own the copyright in the motion picture, it can not make and distribute the film without a license from the book's copyright owner. This is true of software as well. If you write software code and combine it with the software of another, the modified work is a derivative work of the original and you need the permission of the owner of the original software in order to copy and distribute the entire derivative work. However, you can copy and distribute those portions which you independently created, so long as they do not incorporate any of the original code.

The fact that one does not know a work infringes the copyright of another is not a defense to infringement. It may be taken into consideration for the damages awarded, but an unknowing infringer is still an infringer under the law. It is the duty of the user to make sure that they can use copy, publicly perform, distribute, or make derivative works of the software prior to doing so. If you do not have a license, then you should assume you do not have the right to do these actions.

C. OSS and Viral Licenses

Issues related to the use of OSS were discussed in Section I. OSS is sometimes referred to as "freeware" or "shareware" because it can be shared without costs of for a minimal fee. However, it is usually still protected by copyrights, which is how the copyright owner enforces the rules regarding distribution and not allowing users to charge for the program. GNU and similar licenses are often referred to as "viral licenses" because by integrating your software with GNU licensed software, your software must then be distributed under the GNU license as well, which requires source code to accompany any distribution of the derivative work.

D. OSS Increases Intellectual Property Risk

OSS users may not have to pay for OSS when they wish to install it, but they increase the risk that valuable intellectual property rights in proprietary software may be lost if OSS is indiscriminately mixed with proprietary software. Users must ensure that any use of OSS in its products does not compromise either the copyright integrity of user's ownership nor require users to disclose valuable trade secrets in return for using the OSS software. It is not unusual for OSS packages to be composed of components which have many different licenses. Not all OSS carries the appropriate acknowledgments, notices and licenses. There is a very real potential of litigation and damages if the OSS contains unlicensed software. The fundamental challenge with OSS is that one never really knows who owns the copyright, yet only the copyright owner may grant permission to use via a license. Use without license leaves one exposed to infringement risk. These risks should be factored into the decision to use OSS.

E. OSS Risk Mitigation

Given that there are inherent risks involved in the use of OSS, what can a corporation do to mitigate these risks? First, it must train employees that there are risks in using OSS and that the license agreements are important. Engineers often select software based on its technical merits but will give no thought as to whether the license they get with the software will allow them to use it as they desire. Everyone who develops software should be given tools and instructions on how to use OSS.

A process should be established by which engineers wanting to use OSS can review the software prior to use to see if it is suitable. The first step is to find the licenses to use the OSS. Once the licenses have been found, the requestor would be responsible for reviewing them to determine if the licenses will allow use of the software in the manner contemplated. If they appear to do so, the requestor's management must approve the use. After management has approved, a request for review with a description of the manner of use and software licenses is forwarded to Legal. Many times the OSS was developed by an individual or small group and they will include biographies indicating they have or currently are working for a software company. If this is the case, it would be prudent to ask them to state emphatically that they are the owner of the OSS and have the right to grant a license. OSS developers often do not bother to read the license that accompanies the software and therefore do not follow the licensing terms. It would be prudent for companies to conduct an internal review of the OSS with full support of the legal department to: 1) determine what is included in the program and 2) confirm the accompanying license actually grant the rights needed to all the files in the program. This is especially important for software that will be provided to customers or otherwise distributed outside users.

The review process can be time consuming, especially if the software has been modified by many different users over time and the relevant licenses are not included as required. However, this review is necessary because use without a license leaves the user liable for infringement claims. To facilitate these reviews, LM Aeronautics engineering has assigned an experienced software engineer to participate with the Legal Department. It is essential in conducting a good review that the process receives input from a software developer who can dissect the OSS and has knowledge of intellectual property law. It is also important, that the company have a process that includes gates for the release of software that require approval and review prior to any release.

F. Reviewing OSS for Licensing

The OSS development model often envisions software as being developed by a myriad of software developers contributing over time to a software product. Some submit the initial versions to the code project. Others later modify, correct, and improve the code base. All of this is generating an enormous amount of copyrighted material by a plethora of authors. Most of these authors are also employees and their employers may actually own their contributions.

The process of reviewing OSS for licensing sounds simple, but it has the potential to be time consuming. Given that many OSS applications consist of hundreds of files containing source code, each source file must have some form of license contained in or associated with it. Simple tools are available to aid in this process. For Example Microsoft Excel™ may be utilized to work with and retain the information gathered. It can even be used if the open source product is UNIX™ based. All that need be done is unpack the open source product on a Windows system.

The first step is to identify the copyright notices. While a copyright notice is not a license, its presence or absence is significant when inferring licensing. A copyright notice clearly marks the item as copyrighted material. Use without a license could be construed by a court as willful infringement. A tool from Microsoft known as MS Power Toys 95 SendToX is quite useful when coupled with the MS Windows File Search tool to identify the copyright notices and create a list of files with copyright notices.

The next step is to identify the licensing of the files. Typically an OSS program file itself will contain a short license, which licenses that file. The file will contain a reference to a known license, for example the GNU GPL. If the file contains a reference to a license file by file name, the licensing is direct and clear and will be graded low risk. Failing to find direct licensing then leads one to find an inference to a license elsewhere in the program. Licensing is inferred by linking the file in question to a license by proximity of a license to the file in question. In general, license files found in sibling directories are not inferred as the licensing of a file in question but rather a current or parent directory that has a license connected to the root directory of the open source. The one exception that is permitted is a license file found in a documentation or license directory. It is usually necessary to dissect an open source product to determine how it is licensed. Again the SendToX tool and File Search tool can be utilized.

The next step is to merge the inventory, the copyright notices, and licensing together into a manifest so that for each file you have its name, copyright notice (if present), and licensing. MS Power Toys 95 SendToX will let you send the names of a group of selected files to the clipboard. That list of names can then be pasted into an Excel worksheet. The File Search tool can be asked to produce a list of all the files in a directory structure and by selecting all the files, the names can be sent to the clipboard. Thus you have an inventory.

After creating the manifest, all files, which are directly licensed, are graded low risk. Low risk means that licensing of the item is clear. All files which are inference licensed are graded medium risk. Medium risk means that licensing of the item is being inferred but the inference of licensing could be incorrect and thus some risk remains if the assumptions are incorrect. All remaining files are graded high risk. High risk means that licensing of the file could not be determined. After grading for infringement risk, management is advised if there are high-risk items that must be removed prior to use and to decide whether to assume the risk associated with the medium risk items.

G. Costs of OSS License Review

Reviewing open source can be quite laborious. For example, an open source product which has the same copyright notice in each file and each file references a known license can be processed in less than 1 hour of labor, even if it contains thousands of files. A different example is a 5,000 file open source product where each file contains a unique copyright notice and self licenses with varying types of license text, which will require about three minutes per file to extract the copyright notice, record and read the license. That equates to 250 hours of labor. At an engineering labor rate of $100/hour the product review costs approximately $25,000. The vast majority of OSS requires 8–16 labor hours to review. This equates to a cost of between $800 and $1,600 for the engineer, but there are the additional costs for engineering review, security and legal review, configuration management (CM), cost, etc. associated with using "free" open source.

III. OSS in Safe and/or Secure Systems

Assuming a company has made the determination to proceed with usage of OSS after reviewing and accepting all the risks regarding copyright and licensing, there are additional cost metrics which must be considered if a team intends to use a product within a safety critical system or in a role in which Information Assurance is a requirement. Such assurances are actually interrelated (as discussed below in Section D). Individuals commonly use the terms interchangeably. For example one might query, "How safe is your home?" An appropriate response must be predicated on the threat and assumed probability of occurrence or exploitation. Are they asking if there are sufficient mechanisms such as fire, smoke, or carbon monoxide detectors or if one has a security intrusion detection system? Generally when the threat agent is an "engineering oversight (error)", natural act, or "accident" one is using "safe" in the correct sense. If the threat agent is a malicious individual breaking into the home, then the more correct verbiage would be to ask, "How secure is your home?"

The categorization of expected threats and potential exploitations is a part of the software engineering process for those involved in developing either safe or secure software. Based on the risk analysis one applies risk mitigation techniques to eliminate or reduce the potential damage in the event of a breach. The software assurance evidences are amazingly similar whether talking safety or security.

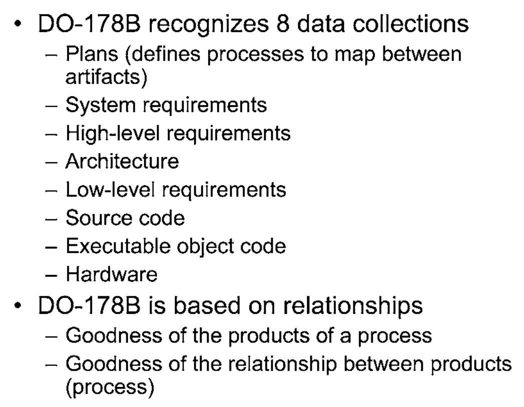
A. Safety Critical Usage of OSS

Lockheed Martin Aeronautics requires the use of safety critical processes and products associated with our avionics. For civil aviation, the design, implementation, and testing of safety critical software falls under the purview of the Federal Aviation Agency (FAA). They use a standard to establish the assurance certification known as RTCA DO-178B.[1] (NOTE: There are other international standards for aviation as well that are reciprocally recognized by the various countries.) Outside of aviation there are other safety standards which apply to software development, such as for medical devices and nuclear power plant controls.

The DO-178B standard for safety certification requires a stringent design discipline both from a procedural and process standpoint as well as in the actual considerations of the functional design of the code base itself. There has to be extensive evidence showing clearly traceable mapping from requirements to high level design to low level design to implementation. Software tools have to be validated (not certified) which includes compilers as well as other modeling tools. Testing and test scripts, suites, etc. must derive from requirements to test the final implementation. In addition there must be modified condition decision coverage testing which goes far beyond normal software development testing.

It is not within scope of this paper to educate the reader on the development of safety critical software. A generally accepted metric requires a solid software engineering organization with an SEI CMM Level-3 "like" process in place for software development. When compared with good non-FAA software processes, in order to meet the additional software "assurance" requirements under DO-178B, one would incur about 1.5 to 3.0 times the effort and cost to develop software with the necessary assurance artifacts to satisfy FAA reviews. This is based on the assumption that a team is starting from scratch. If one starts with pre-written code, such as an OSS, the issues are compounded. Several questions must be asked.

1. Was the code developed by competent programmers that understand safety critical software techniques?
2. Can the documents for the requirements, design, formal code reviews, etc. be downloaded along with the source code?
3. What is the evidence for solid, documented configuration management of the code base throughout its development cycle?
4. Are any test suites available?
5. Are there code sections in the baseline that are simply not assurable?
6. Is there dead code? (Dead code is code that cannot be traced back to a high or low level requirement. It is not allowed in DO-178B certifications. There is a difference between dead code and de-activated code. However one must be able to clearly show that the de-activated code is truly "off-line" and cannot activate in flight.)



- DO-178B recognizes 8 data collections
  - Plans (defines processes to map between artifacts)
  - System requirements
  - High-level requirements
  - Architecture
  - Low-level requirements
  - Source code
  - Executable object code
  - Hardware
- DO-178B is based on relationships
  - Goodness of the products of a process
  - Goodness of the relationship between products (process)

Fig. 1 DO-178B in a Nutshell.

As shown in Fig. 1, FAA safety critical assurances require a top to bottom traceability from highest level design documents down to the executable code on the computer hardware. Basically there must be evidences that a proper correspondence (or relationship) exists between the eight data collections as shown. Quality Assurance is a mandatory part of the entire process requiring constant overview and supervision. The "certification package" is comprised of a host of documentation describing to the certifier how the software was designed, developed, tested, etc. In other words, the certifier has to be convinced by the evidences that the builder built it correctly to do what is was intended to do and nothing else. This discussion is not intended to cover all the issues but to offer the reader a sampling of the kinds of things that go into a safety critical software engineering effort.

For an OSS product, one must anticipate that a development team would expend extensive labor to create evidences covering design reviews, documentation, testing, etc. A team attempting to produce the mountain of assurance data necessary to adequately convince a certification official of the safety assurances of the OSS software after the fact would indeed be a momentous task and *quite expensive*.

What may be difficult to scope ahead of time is that one may simply run into problems with "legacy code" that are not certifiable post facto. The biggest obstacle for a post facto effort is that in some cases the development of the safety relevant artifacts must occur *at the time of development within the established process*. It could be that the team would have to re-write some sections of the code under the software assurance process. This would also be the case with bullet #1 above as some functionality will likely be implemented in a way as to actually be unsafe. If such rewrites have been accomplished, would the developer now be compelled under the licensing agreements discussed in section II above to make such changes freely available to the OSS code base? Would your company be willing to do so? How does that affect your cost model?

B. International Common Criteria and Information Assurance

The International Common Criteria (CC)[2] information assurance (IA) standards were designed to provide a common framework for security evaluation of Commercial Off the Shelf (COTS) products that have Information Assurance requirements. Interestingly, while the original focus of security evaluation criteria[3] concerned Department of Defense and U.S. Government computer systems, such products have great utility to many commercial IT departments.

The certification of these products is handled by the Common Criteria Evaluation & Validation Scheme (CCEVS) which is administered by the National Information Assurance Partnership (NIAP). NIAP is co-sponsored by the National Institute of Standards and Technology (NIST) and the National Security Agency (NSA). NIAP certified laboratories are for-profit organizations which are hired by COTS vendors to provide services leading to certification of products at varying levels of assurance. The levels are known as Evaluation Assurance Levels 1 through 7, level 1 being the lowest assurance. The higher the level of assurance the more rigorous and, and hence the more expensive the "proofs" of assurance become.

From an individual CC partner country's point of view, because of the international mutual recognition agreements of Common Criteria, COTS products certified at EAL 1 through EAL 4 under CC equivalent processes can be used in IA environments by other partner countries without need for review. This mechanism allows for potential reuse (via COTS sales) to amortize the costs of certification. In the United States the NIAP labs can provide this certification service. However at the higher assurance levels, the labs must coordinate with NSA in the evaluation and certification process. For more information on the process and program see www.niap.nist.gov/cc-scheme.

While the CC was initially targeted at COTS, it is equally applicable to OSS as well. The business model for OSS becomes a little problematic. If a commercial vendor designs, builds, and markets a product, then proceeds with a NIAP evaluation at some assurance level, that vendor receives a certificate based on a specific reference implementation. The vendor is well aware of the costs of evaluation and if it chooses to release a new version, the expense of doing a re-evaluation is part of the entire business plan decision. Since OSS is owned by no one in particular, who pays for the evaluation? Assuming an IBM decided to evaluate a specific version of LINUX on a specific hardware platform, known in CC terms as a Target of Evaluation (TOE). There is no mechanism for H-P to claim NIAP evaluation on H-P hardware. They would have to fund their own evaluation. These "business case" issues aside, the point is that NIAP evaluations can be done on OSS as well as commercial products.

C. U.S. Government Security Policy

On the safety critical side, the FAA not only participated in the development of the RTCA DO-178B standard, it required all aviation industry companies to adopt it. Thus standards and legal mandates were established to clearly enforce the execution of the program. Even though the CC / NIAP process was in place, there was no ground swell of vendors proceeding with evaluations. Basically there was no "market pull", mandate, or requirement for evaluated products, except for a few specialized NSA programs. In order to drive the commercialization and commoditization of the use of NIAP evaluated COTS products, an overarching policy or mandate was necessary.

The National Security Telecommunications and Information Systems Security Committee (NSTISSC) was established by National Security Directive (NSD) No. 42, dated July 1990, and is responsible for developing and promulgating national policies applicable to the security of national security telecommunications and information systems. The committee (now known as the Committee on National Security Systems: CNSC) issued a National Information Assurance Acquisition Policy known as NSTISSP #11.[4] The policy, initially issued in January 2000 and revised in July 2003, uniformly mandated to all U.S. Government departments and agencies (including DoD) some sweeping requirements with respect to COTS products and IA. The first was the definition of IA requirements.

(1) IA shall be considered as a requirement for all systems used to enter, process, store, display, or transmit national security information. IA shall be achieved through the acquisition and appropriate implementation of evaluated or validated GOTS or COTS IA and IA-enabled IT products. These products should provide for the availability of the systems, ensure the integrity and confidentiality of information, and ensure the authentication and non-repudiation of parties in electronic transactions.

The second was the mandate to certify such IA products under the Common Criteria (or equivalent) as follows.

(2) On 1 January 2001, preference was to be given to the acquisition of COTS IA and IA-enabled IT products (to be used on systems entering, processing, storing, displaying, or transmitting national security information) which had been evaluated and validated, as appropriate, in accordance with:
- The International Common Criteria for Information Security Technology Evaluation Mutual Recognition Arrangement;
- The National Security Agency (NSA) /National Institute of Standards and Technology (NIST) National Information Assurance Partnership (NIAP) Evaluation and Validation Program; or
- The NIST Federal Information Processing Standard (FIPS) validation program.

(3) Effective 1 July 2002, the acquisition of all COTS IA and IA-enabled IT products to be used on the systems specified in paragraph (2), shall be limited only to those which have been evaluated and validated in accordance with the criteria, schemes, or programs specified in the three sub-bullets of paragraph (2).

D. Harmony Between Safety and Security

There is an interesting synergy between safety and security which fits well for the aerospace industry in particular. The qualities needed for high assurance safety, such as in a DO-178B Level A certification, form a very good baseline for high robustness security. Both deal with confidence or assurance of the quality of the product (remember DO-178 is about assurances, not just a solid process for developing software). In a study funded by the NSA, University of Idaho's Center for Secure and Dependable Systems completed a study mapping the fit between a DO-178B Level A assurance and a CC assurances. In that study[5,6,7] it was demonstrated that if one has a software product that has achieved high safety assurance, the product not only has the functional and developmental quality necessary for NIAP EAL-4 but also the necessary assurance evidence as well. By using a product previously evaluated at DO-178B Level A (an equivalent to EAL-4) one can actually achieve evaluation of that product at EAL-7 affordably.

Starting with a DO-178B Level A baseline, the additional assurances include semi-formal / formal modeling, mathematical proofs of state changes, and a host of other "assurance documentation". Lockheed Martin Aeronautics and Green Hills Software (GHS) partnered for the initial EAL-6+ (or 6 Augmented)[§] NIAP evaluation of the GHS INTEGRITY-178B RTOS. GHS provided data shows that 3 to 3.5 man-hours per line of source code was expended. There was a previous investment of about 2 to 2.5 man-hours per SLOC to create the DO-178B Level A compliant assurance evidences. As noted above DO-178B has to be integrated as part of the software process. The hours noted here are for the assurance work and should not be considered the sum total of all the software development effort. In addition once the COTS vendor has performed the initial evaluation, early data suggests that long term NIAP re-evaluation costs for new releases can drop by 80% or more. Therefore a security management team can realize affordable reuse by performing NIAP evaluations of existing COTS safety products that have widespread commercial support such as real-time operating systems (RTOS), Middleware, Network devices, etc. This component aggregation of security layering is known as the Multiple Independent Levels of Security Architecture.[8,9,10]

In all these cases, the size of the code base is much smaller than, for example, general purpose operating systems (OS's). The guidelines are that to do these higher level evaluations which rely on formal models, one must have a code base that is small (4K to 10K SLOC). Most formal methods experts agree that once a product gets above 15K SLOC, the proofs are intractable, no matter how much one may be willing to spend. As it turns out DO-178B Level A, RTOS's actually are quite small. In addition they provide a great deal of functionality, sufficient to develop and execute 2 Million SLOC applications so such products are not as restrictive as many would think.

Alternatively, if one did not have the DO-178B baseline product as a starting point for security evaluation, the evaluator indicated that a minimum figure of $5 Million per 1000 SLOC should be expected to develop the evaluation package for EAL-6/7. At the medium robustness level (EAL-4) the costs "from scratch" for the evaluation package are between $50K and $150K per 1000 SLOC. It should be noted that medium robustness products are generally not acceptable in separating more than one level of security data, hence terming such products as MLS capable would not be entirely true. However there is need in networked architectures for these EAL-4 evaluated products as one aggregates the entire system.

E. Security Assurance and OSS

As with the Safety Critical discussions some questions need to be addressed concerning OSS for security assurance.

1. Was the code developed by competent programmers that understand security functional software techniques?
2. Can the documents for the Security Policy, the Functional Requirements, and the Assurance Requirements be downloaded?
3. Can the evidences showing traceability of the code back to the items in number 2 above be downloaded?
4. Can one download solid, documented configuration management of the code base throughout its development cycle to prove to an accreditor that the code downloaded and being used was not tampered with in delivery to the user? For example, how can one provide reasonable assurances that the LINUX Kernel Linus Torvalds released has not been modified on one of the sites a user chooses. (One assumes that a major integrator can provide evidence of trusted delivery to the end user but how about before the integrator "downloads" OSS to begin use?)

At the higher assurance levels (EAL-5 through 7), from the standpoint of the NSA and the U.S. Government there are additional requirements on not just the quality of the software but the pedigree.[11]

1. In what country was it designed and built?

[§] EAL-6+ is the CC equivalent to DoD "High Robustness" software

2. What is known about the nationality and therefore the security risks of the individuals contributing to the code base?
3. Is there subversive code in the product?

It is these simple "background" issues having nothing to do with the quality or evaluatability of the product from its functional or performance standpoint that can derail usage of OSS in high robustness security (EAL-6/7) systems. If the data on the developer's "national pedigree" is not freely available, the user of an OSS product would have to invest an enormous amount of manpower to review every line of code. Companies using OSS would need individuals specifically trained in security techniques to spot malicious code deliberately inserted by un-trusted individuals in the implementation process. Known as subversive code, it is not easy to find these and remove them after insertion. In fact it may be impossible to achieve after the code is developed.[12]

While commercial products are not automatically free from the issue of subversive code, there are techniques that can lead to risk mitigation that are not possible in the OSS community. The best way to deal with insertion of subversive code is to prevent the insertion in the first place during the engineering process. Such measures include but are not limited to, a good CM process controlling access to and logging code checkout by user id, two person code reviews prior to CM code check in, evaluation / security monitoring teams separate from design / implementation teams, and a secure code build process and delivery. Commercial companies undergoing a NIAP evaluation are required to address the personnel pedigree, trusted build, and product delivery process assurance requirement activities. Since they have complete control of the product and its development lifecycle, it is achievable. It becomes much more problematic with OSS.

Certainly commercial companies can potentially achieve the same thing using an OSS product and charge for the value added under GNU / GPL. However, using the manpower figures for NIAP evaluation at EAL-6/7 as indicated in section D above, *then* adding the additional costs in determining and validating the national background of the various developers which contributed to the OSS, *and* finally funding a line by line scrub of the code to determine if subversive code is embedded in the source files, one has to conclude it becomes highly problematic that a business case can be made. However, assuming that a business case could be made for that kind of investment, such a delivery to a customer would not be the same as that customer downloading an OSS product from some random site on the internet. Obviously the customer would have to pay a hefty fee to the seller as part of the value added. Also the costs of a trusted distribution would not be "free" either. Such a business model really begins to diminish the reality that NIAP evaluated OSS based secure implementations are any more cost effective than a NIAP evaluated commercial proprietary products at the EAL-4+ and above levels. There is the potential of using OSS for products at the EAL-1 through 4.

Having made the decision to use, evaluate, and distribute an OSS based product, a company would have to plan for version upgrades based on whatever schedule that OSS community wishes to release. At each release the company would have to repeat everything previously done since additional world wide contributors could have added new code (pedigree) and with the download the possibility for new subversions. While this may be acceptable, it will not be inexpensive, for in reality that company will always have a proprietary, configuration managed version of the OSS product to maintain and sell to its customer base.

IV. Conclusion

Hopefully it has occurred to the reader that, as indicated in this paper's title, "Free Is Not Exactly Inexpensive", the choice to use OSS has to be considered carefully early in the project. Such a decision will have to include full support and review by IT buyers and legal counsel to insure license compliance. Labor hours will need to be expended after download to insure all OSS licensing restrictions are in compliance. Further in some high safety and/or security systems, such as those found in aerospace, there will be a very large engineering effort required which adds substantial costs to the *free software*. In these environments the potential user of OSS will want to investigate the use of COTS products based on open standards for use in safe / secure environments where the vendor has already accomplished the investment to provide the certification documentation as well as the licensing rights. It may be the much better alternative. Open source may be acquired at no cost by simply downloading it from the internet. The bottom line is that the use of open source software for product development has hidden costs to confirm licensing rights and very large hidden costs for use in safe and/or secure environments. In any case, it is not free.

References

[1]RTCA DO-178B, “Software Considerations in Airborne Systems and Equipment Certification”, Copyright 1992, RTCA, Inc., 1140 Connecticut Avenue, N. W., Suite 1020, Washington, D.C. 20036

[2]“Common Criteria for Information Technology Security Evaluation”, September 2006, Version 3.1, CCMB-2006-09-001, 002, 003.

[3]Department of Defense, “Department of Defense Trusted Computer System Evaluation Criteria”, December 1985, DOD 5200.28-STD (supercedes CSC-STD-001-83).

[4]NSTISSP No. 11, Revised Fact Sheet, National Information Assurance Acquisition Policy. Available at http://www.cnss.gov/Assets/pdf/nstissp_11_fs.pdf

[5]Alves-Foss, et al. (2001), “Toward Common Criteria Certification for DO-178B Compliant Airborne Software Systems”, University of Idaho. Available at http://www.csds.uidaho.edu/papers/Alves-Foss02b.pdf.

[6]Taylor, C., Alves-Foss, J., and Rinker, B., “Merging Safety and Assurance: The Process of Dual Certification for FAA and the Common Criteria”, *Software Systems Technology Conference*, April 2002.

[7]Jim Alves-Foss, Bob Rinker, and Carol Taylor, “Executive Summary: Towards Common Criteria Certification for DO-178B Compliant Airborne Software Systems, “*High Confidence Sofware and Systems*”, March 2002.

[8]Vanfleet, Mark, W., Calloni, Ben, A., Ph.D., et al., “MILS: Architecture for High-Assurance Embedded Computing”, Crosstalk Magazine, August 2005, pp. 12–16.

[9]Vanfleet, Willard Mark, et al., “An Architecture for Deeply Embedded, Provable High Assurance Applications”, May 2003.

[10]Rushby, John. A Trusted Computing Base for Embedded Systems. *Proceedings of the 7th Department of Defense/NBS Computer Security Conference*, September 1984: pp. 294–311.

[11]GAO Report, “DEFENSE ACQUISITIONS: Knowledge of Software Suppliers Needed to Manage Risks”, GAO-04-678, July 2004.

[12]Anderson, Emory A., Irvine, Cynthia E., and Schell, Roger R., “Subversion as a Threat in Information Warfare”, Naval Post Graduate School Thesis. Available at http://cisr.nps.navy.mil/downloads/04paper_subversion.pdf

Christopher Rumsey
Associate Editor