Federal Financial Institutions Examination Council

3501 Fairfax Drive • Room 3086 • Arlington, VA 22226-3550 • (703) 516-5588 • FAX (703) 516-5487 • www.ffiec.gov

October 21, 2004

Risk Management of Free and Open Source Software

Purpose

This guidance is intended to raise awareness within the financial services industry of risks and risk management practices applicable to the use of free and open source software (FOSS).[1] For the purpose of this guidance, FOSS refers to software that users are allowed to run, study, modify, and redistribute without paying a licensing fee. Access to source code is a pre-requisite to the use of FOSS.[2] A few of the most well-known examples of FOSS are the Linux operating system, Apache web server, and mySQL database. FOSS is also widely used for network monitoring, diagnosis, and vulnerability testing tools such as the Snort and Kismet network intrusion detection systems, Nessus and Nmap security scanners, and Kismet wireless network detector.

The Federal Financial Institutions Examination Council (FFIEC) agencies[3] believe that the use of FOSS by financial institutions or their technology service providers (hereafter referred to as institutions) involves strategic business decisions. The implementation of those decisions should include prudent risk management practices.

Introduction

The use of FOSS is increasing in the mainstream information technology (IT) and financial services communities. The agencies believe that the use of FOSS does not pose risks that are fundamentally different from the risks presented by the use of proprietary or self-developed software. However, the acquisition and use of FOSS necessitates implementation of unique risk management practices.

Institutions should continue to refer to the risks and risk mitigation strategies outlined in the *FFIEC IT Examination Handbook*, "Development and Acquisition Booklet" (D&A Booklet).This guidance supplements the D&A Booklet by addressing strategic, operational, and legal risk considerations in acquiring and using FOSS.

Strategic Risks

Software requirements should be driven by the institution's strategic business objectives. Institutions should evaluate the benefits of implementing software in terms of its effectiveness, efficiency, and ability to support future growth. Key risk management considerations include code customization, IT architecture, product maturity, forking[4], systems integration and support, and total cost of ownership.

Ability to Customize

Because FOSS source code is publicly available, institutions have the opportunity to modify the software to better align IT capabilities with business strategies. Software modification presents risks similar to self-developed code, and those risks should be addressed in a similar fashion. The institution should test the revised code to ensure performance and the maintenance of confidentiality, integrity, and availability of systems and data. The institution should carefully consider its technical and legal ability to modify and maintain the code, and ensure that controls are in place to protect against copyright and patent infringement.[5]

Compatibility and Interoperability[6]

Typically, proprietary software products from the same vendor (e.g., a product suite comprised of an operating system and applications) are certified to be compatible with one another. In addition, smaller software vendors may create specialized applications in cooperation with a larger vendor so that the application is compatible with a particular operating system or other application with which it must interface. FOSS is often written to open standards[7] and is generally more interoperable than proprietary software. However, the interoperability of FOSS programs may not be formally certified. Therefore, institutions using FOSS should exercise due care to ensure it meets their needs for compatibility and interoperability. An institution may need to augment its IT skills, either internally or by employing a person or firm trained in software integration, to integrate FOSS successfully into its operating environment.

Maturity

Institutions should consider the maturity of any software considered for use in the production environment, particularly for mission critical applications. Mature software generally presents fewer risks than less mature software. Because FOSS development is fundamentally different from other software development, the relevant indicators of maturity may differ. Some factors to consider when assessing the maturity of FOSS are:

- How long has the software been supported or in use?
- How is the development community organized and how well does it function?
- How active is the development community?
- How much published material is devoted to the software?

- How many commercial vendors support the software?
- What is the security track record of the software?

Typically, mature FOSS has large and active development communities, with a project lead determining which new or modified code is incorporated. Additionally, increasing numbers of commercial vendors now support mature FOSS.

Forking

Forking is of particular concern in the FOSS development process. A fork occurs when the development community splits over the path of development of a given application. In the worst-case scenario, development of forked FOSS may be halted, or the technical direction may become so altered that it no longer meets the institution's needs.

Institutions should mitigate this risk by ensuring that adequate support is available for the current FOSS software either in-house, through vendors, or other outside sources.

Systems Integration and Support

FOSS can be acquired and implemented with varying degrees of integration and support. For example, an institution may obtain FOSS from a systems integrator that ensures the compatibility of all FOSS components. Conversely, an institution may obtain FOSS directly from multiple development projects and integrate the components in-house. Integration includes the initial implementation of the FOSS as well as subsequent maintenance and upgrades. Institutions that choose to integrate FOSS in-house should carefully consider their ability to identify, track, evaluate, appropriately modify, install, and maintain the software.

Proprietary software vendors may compel institutions to upgrade to the newest version of their software or risk discontinuation of support. In contrast, since institutions have access to the FOSS source code, they can extend the software's useful life through internal or outsourced development and support.

Total Cost of Ownership

Institutions evaluating the total cost of FOSS ownership should include both direct and indirect costs. Direct costs generally include hardware, software licensing, and annual maintenance. One of the features attracting institutions to FOSS is its complimentary or low cost for licensing and maintenance. However, the indirect costs of FOSS may be higher than those associated with proprietary software if existing staff requires more training than would otherwise be necessary with a proprietary product. In addition, change management costs may be higher in a FOSS environment if the institution implements products lacking third-party vendor support. The institution generally will bear more responsibility and spend more resources identifying, selecting, analyzing, and installing upgrades and patches. Depending on the FOSS selected, other indirect costs may appear, such as code reviews, documentation, and contingency planning.

Operational Risks

Operational risks exist within any IT operating environment. Risks, controls, and prudent risk management practices are detailed in several of the *FFIEC IT Examination Handbook* booklets. Operational risk considerations associated with the use of FOSS that warrant attention include code integrity, sufficiency of documentation, contingency planning, and support.

Code Integrity

Code integrity is important when institutions adopt and implement FOSS because the source code is widely available and can be distributed by anyone. Institutions should develop standards and adopt appropriate procedures to ensure that they are acquiring the source code from a trustworthy party, and they should verify the integrity of the code they receive. The same standards and procedures should apply to subsequent software updates and patches. Once an institution has established that the party is trustworthy, a variety of techniques, such as PGP[8] encryption and MD5[9] hash comparisons, should be used to validate the authenticity and integrity of the code. Institutions can also have internal staff review source code to verify its integrity. However, such reviews can be time consuming, require considerable technical expertise, and may not identify all issues.

Sponsoring organizations, such as SourceForge (sourceforge.net), can provide some assurance to users that they are downloading unadulterated code. For information on security advisories and incidents affecting the integrity of downloaded open source code, users can reference the CERT® Coordination Center (www.cert.org) or other reputable security industry organizations.

Documentation

The documentation that accompanies FOSS may be less comprehensive than the documentation that accompanies proprietary software because of the diversity of the development community (i.e., supporting organizations, third-party vendors, and individual developers). Institutions should ensure their software acquisition policies delineate minimum documentation standards and establish procedures for supplementing inadequate documentation.

Contingency Planning

The continued viability of FOSS is largely dependent on the open source community and third-party vendors. Institutions using FOSS can end up with "dead-end software" if the development community abandons a product. Other outside forces, such as unexpected litigation, may also compel an institution to terminate its use of a particular FOSS application.

Institutions should mitigate these risks by ensuring that adequate support is available for the current FOSS software either in-house, through vendors, or other outside sources, and developing an exit strategy for replacing mission critical applications.

External Support

External support for FOSS is becoming more robust. FOSS users are no longer as dependent on informal support, such as the FOSS development community and Internet mailing list. These resources still exist, but the entrance of value added resellers (VARs) and independent vendors of FOSS support services now provide a wider range of choice for FOSS users.

The maturity of certain FOSS projects, such as the Linux operating system, has motivated VARs and independent support vendors to enter the market with FOSS business solutions. Institutions should understand that these firms

- May offer comparable support and service levels to those offered by traditional propriety resources.
- Are increasingly beginning to offer support for FOSS systems and applications that are no longer supported by the open source community or vendors, allowing institutions to retain the use of particular FOSS systems and projects for extended periods of time.
- Are becoming more numerous, thus providing institutions with more options in choosing support appropriate for their unique expertise and operating environments.

When evaluating support from the FOSS development community, institutions should

- Have available highly competent expertise to be able to differentiate between reliable and questionable open source community support resources.
- Consider participating in formal trade groups, verified government and university programs and projects, and other "business oriented" user communities, rather than generic, widely accessible public online forums.
- Be cautious of elevated reputation risks when using the institution's name or email address in discussions of the institution's operating environment in any public online forum.

Legal Risks

Institutions should identify and consider the legal risks associated with the use of FOSS prior to deployment or development. Key legal risks include licensing, infringement, indemnification, and warranties. In most cases, prior to selecting a FOSS solution, institutions should consult with counsel knowledgeable in the areas of copyright and patent law.

Licensing

FOSS acquisition and use can be governed by any of more than fifty different licenses that have significant differences in the rights and restrictions contained in the license. In general, FOSS licenses permit copying, distribution, and modification of the software, but do not contain any warranty or indemnification. A list of some of the most common FOSS licenses can be found at the Open Source Initiative's Web site (www.opensource.org).

The most common FOSS license is the General Public License (GPL).Software covered by the GPL can be modified, but any release or distribution of modified software must be accompanied by an offer to provide the source code under the same GPL license. Stated another way, anyone can use the software and change the program code, but the new code cannot be redistributed as a proprietary application.

The Berkley Software Distribution license (BSD) is another common FOSS license. It also allows redistribution of source code, but with a few basic restrictions. For example, the code must retain a copyright notice and disclaimer and a stipulation that the entity providing the license is not to be used for endorsements of derivative products. However, the BSD license does not include a clause requiring a specific licensing model for derivative works. This allows products created using BSD-licensed code to be used in proprietary software.

The terms and conditions of proprietary software licenses typically require a seat management program where users and available licenses are tracked and matched to avoid violating the terms of the license agreement. Customarily, FOSS does not license by seat, which may result in significant cost savings. In some cases, FOSS sold by VARs may have a license fee based upon the number of servers on which the software is installed.

Institutions considering the use of FOSS should seek qualified counsel regarding the requirements and restrictions of the particular license governing possession and use of the software. Institutions should be aware of the fact that FOSS usage may not require the execution of a traditional written contract. In most cases, the electronic download agreement or mere use of the code binds the institution to the terms of the license. Institutions should be prepared to demonstrate they performed a legal review of FOSS licenses, track licenses and changes to them through automated or manual means, and understand the legal consequences of combining open source and proprietary software.

Infringement

Institutions that use computer software run the risk of being sued for either copyright or patent infringement. However, the potential for an infringement lawsuit is more likely if the institution is using FOSS because, unlike proprietary software, FOSS is developed in an open environment where code is shared and modified by numerous unaffiliated parties. This code sharing increases the possibility that proprietary code may be inserted in the FOSS at some point during the development process. Institutions can mitigate this risk by

- Retaining qualified legal counsel to advise the institution concerning FOSS licensing.
- Implementing enterprise-level policy and business rules that mandate strict adherence to license terms and conditions.
- Using automated tools to track licenses and changes.

- Understanding the consequences of combining FOSS and proprietary software.
- Evaluating the strength of any indemnities.
- Developing contingency plans that will allow the institution to continue operating even if infringing code is taken out of production.
- Using a control mechanism to ensure that all code contributed to FOSS projects is original and written onsite, such as a "clean room."[10]

Warranties and Indemnities

Proprietary software licenses customarily include a warranty that the software will achieve a specified level of performance and an indemnity that the vendor will defend the user in the event of an infringement lawsuit. In contrast, FOSS is customarily licensed "as is," without warranty or indemnity. Recently, VARs have begun to market FOSS with dual licenses. The first license is usually some form of the GPL, and it covers the rights and obligations associated with the use of the software. The second license describes the support services to be provided by the VAR and may include performance warranties and indemnities. In some cases, the VAR may agree to support a particular version of the FOSS for a set time. Institutions should evaluate carefully the terms of any indemnification offered by a VAR, as well as its financial capacity to provide a robust defense. Institutions may also consider third-party insurance, if available.

Summary

The use of FOSS by financial institutions does not pose risks that are fundamentally different from those presented by the use of proprietary or self-developed software. However, FOSS adoption and usage necessitates some distinctive risk management practices with which institutions must be familiar. This guidance describes those unique risk management practices and should be used in conjunction with other published guidance, such as the *FFIEC IT Examination Handbook*, Development and Acquisition Booklet.

[1] The use of the word "free" in this context does not necessarily mean that the software is available at no cost. For additional information about FOSS, refer to www.fsf.org and www.opensource.org .

[2] In contrast, users of proprietary software are generally not permitted access to the source code or allowed to redistribute programs.

[3] The FFIEC member agencies are the Board of Governors of the Federal Reserve System, Federal Deposit Insurance Corporation, National Credit Union Administration, Office of the Comptroller of Currency, and Office of Thrift Supervision.

[4] A fork is the redirection of existing FOSS, generally resulting in a new application that may compete with or replace the established FOSS.

[5] Refer to the Legal Risk section for further discussion of these issues.

[6] Interoperability is the ability of a system or a product to work with other systems or products.

[7] Open standards exist to enable interoperability while at the same time ensuring certain minimum requirements are met across diverse hardware and software products and services. For example, the Open Source Development Labs (OSDL) provides computing and test facilities in the United States and Japan to developers around the world. The OSDL is also actively involved in the development and deployment of open source standards.

[8] PGP refers to Pretty Good Privacy, which uses public key encryption to exchange files or messages with confidentiality and authentication.

[9] MD5 refers to Message Digest Algorithm Five developed by Ron Rivest of RSA.MD5 is a one-way hash function that processes input data to create a unique message digest to verify data integrity.

[10] The term "clean room" is a method of writing software whereby developers cannot be accused of reverse engineering an existing product. Briefly, one team studies the behavior and specifications of the product to be copied, and a second team develops the new product without any exposure to the original.