وزارة المواصلات والاتصالات
MINISTRY OF TRANSPORT AND COMMUNICATIONS

Cyber Security Guidelines for Using OPEN SOURCE SOFTWARE

Version: 1.0

Author: Cyber Security Policy and Standards

Document Classification: Public

Published Date: March 2018

Document History:

Version	Description	Date
1.0	Published V1.0 document	March 2018

Table of Contents

Legal Mandate(s)

Emiri decision No. (8) for the year 2016 sets the mandate for the Ministry of Transport and Communication (hereinafter referred to as "MOTC") provides that MOTC has the authority to supervise, regulate and develop the sectors of Information and Communications Technology (hereinafter "ICT") in the State of Qatar in a manner consistent with the requirements of national development goals, with the objectives to create an environment suitable for fair competition, support the development and stimulate investment in these sectors; to secure and raise efficiency of information and technological infrastructure; to implement and supervise e-government programs; and to promote community awareness of the importance of ICT to improve individual's life and community and build knowledge-based society and digital economy.

Article (22) of Emiri Decision No. 8 of 2016 stipulated the role of the Ministry in protecting the security of the National Critical Information Infrastructure by proposing and issuing policies and standards and ensuring compliance.

This guideline has been prepared taking into consideration current applicable laws of the State of Qatar. In the event that a conflict arises between this document and the laws of Qatar, the latter, shall take precedence. Any such term shall, to that extent be omitted from this Document, and the rest of the document shall stand without affecting the remaining provisions. Amendments in that case shall then be required to ensure compliance with the relevant applicable laws of the State of Qatar.

Acronyms

GNU GPL	GNU's Not Unix General Public License
OSS	Open Source Software
TCO	Total Cost of Ownership
SLA	Service Level Agreement

Introduction

"Source code" within the context of computer applications are the list of commands / scripts put together to create an application. It is the part of software that most computer users don't ever see.

Any changes to the logic, feature sets or look and feel of the application is achieved by modifying the "program's source code". Usually this code is a guarded secret by application developers and is the developers / organization's Intellectual Property, often secured through Intellectual Property Rights, Copyrights etc.

Open source software is a software, where its developers (individuals and/or organizations) make the source code available to public so that anyone can inspect, modify, enhance it and redistribute it. It is generally distributed under special licensing schemes like Common Creatives (CC), GNU General Public License (GNU GPL) etc.

Open Source Software present several benefits as well as challenges, these are covered in additional details below.

Objective

The objective of this guideline is to provide security guidance to our stakeholders while choosing open source software solutions. It will help organizations to understand and evaluate the security risk associated with using Open-Source Software and how to mitigate it.

Scope and Audience

Any individual or an organization that is using or is considering using Open-Source software within their operations.

Guidelines

The decision of whether to choose between open source and proprietary software involve significant complexity and goes beyond mere price factors. Organizations should take an informed decision and consider amongst others, the following factors prior choosing an Open Source Software

a. **Total Cost of Ownership (TCO):** Open-Source does not necessarily mean free. Although the license costs are zero, there may be charges for additional support that organizations may need to have, fixing bugs, trainings to develop skills within the team, premium that organizations may need to offer to hire employees with such skills etc.
    a. **Usability:** OSS often is not user friendly, as the developers focus on features rather than usability.
    b. **Skilled Resources:** Organizations may face challenges in hiring or retraining their existing staff on skill sets needed to support and develop the OSS in-house.

b. **Roadmap for Intended Usage:** Organizations may choose to use the product as is or have the option to modify it to meet their business needs since the source code is available. Alternatively, organizations could also use the source code as a base to flesh out a complete new development thereby allowing them to reduce their development time. However, organizations need to evaluate the license of source code, as certain open source usage license mandate that the modified code is published under similar license.
c. **Professional tech support**: OSS is usually a community initiative / open forum, nobody is obliged to help you. Organizations may have to do / fix things on their own or wait for responses from the OSS community and cannot expect time bound SLAs. However, professional technical support is available for certain OSS initiatives / products like LINUX.
d. **Security:** It is dangerous to assume that OSS is implicitly secured and safe to use. However, since the source code is in public domain, the code is up for review by anyone and security enthusiasts review such codes to identify vulnerabilities within the code. Multiple security researchers and organizations (Multi Eye Principle) review the code and look at it from their individual perspectives. Further, since the code is in public domain, the risks of it having Trojans is much less. Nevertheless, due to a lack of central oversight / authority that provides quality / security assurances on the code that is distributed it is possible that malicious users may embed or corrupt the code with malwares / Trojans since the code is publicly available. Moreover, potential exploits (Proof of concepts and ready to use) may be readily available on the internet for known and published vulnerabilities in such OSS.

Once an organization has made a decision to deploy and use OSS, they should consider the following security controls to provide a layer of assurance to their business.

1. **Governance:** Organizations should develop a comprehensive policy governing the usage of OSS. Amongst other things, the policy should cover an acceptable usage of OSS and the acceptable risk appetite for OSS. Risk Assessment should on a minimum cover risks related to license requirements, operations (support for the software and stability of the software) and security (vulnerabilities and known exploits).
2. **Asset Inventory:** Organizations should maintain a detailed inventory of OSS used within the organization detailing instances (number / quantity) of use, version etc. Where OSS is used as a component of your in-house application, a data call should be performed to determine what components are OSS and what versions are currently is use. The inventory should include libraries, frameworks, middleware and applications. Maintain a profile of each OSS to include the code's origin, where to get updates, and how often the community releases new versions
3. **Documentation:** Where OSS is used as a component of your in-house application, a comprehensive documentation of the components used should be maintained. This includes libraries, frameworks, middleware and applications.
4. **Source Code Repository:** Maintain a repository of the source code of OSS deployed within the organization.
5. **Whitelisted OSS:** OSS should be qualified after conducting relevant usability, stability and security tests. Only pre-approved and qualified OSS should be used and deployed within the organization.

6. **Baseline Controls:** The OSS should be treated as any other information system deployed within an enterprise and should comply with the entire relevant baseline and recommended controls prescribed within the National Information Assurance Policy including its classification.
7. **Risk Management:** Organizations should conduct a Risk Assessment on the OSS, based on the criticality of the services that it will deliver and implement necessary controls to manage any risks that may arise.
8. **OSS Installation:** Any OSS installed / deployed should adhere to the organization's system installation procedure. This should ensure that:
    a. Only whitelisted OSS is deployed in the organization.
    b. All deployments are vetted and approved through a formal system deployment / change management process.
    c. All deployments are inventoried in the asset register.
    d. Only authorized individuals such as the system administrators should install / deploy OSS.
    e. Use OSS from reliable and trusted sites. Wherever possible prefer source code to binaries. It is always good to download the source code, verify against the MD5 checksums provided. Examples of trusted sites as recommended by Open Source Initiative include freshmeat.net, sourceforge.net, osdir.com, developer.berlios.de and bioinformatics.org
    f. Ensure that the OSS is tested and updated with the latest patches.
9. **Security:** Perform a security assessment to identify and patch any known vulnerabilities in the OSS. For critical applications, it is recommended to do a combination of an automated static analysis (source code scanning) and dynamic analysis to find vulnerabilities in individual applications. On identifying a vulnerability the organization should:

    a. Check if a patch or an updated version is available to fix the identified vulnerability.
    b. With caution and responsible disclosure, ask the open source community for help. Post the issues to the community and see if a member can help with the fix.
    c. Fix them yourself. Use your own or third-party development resources to resolve the issues.
10. **Application Hardening:** As with any other software, the OSS should be configured in a secure manner.
    a. Disable unwanted services.
    b. In a development environment, unused and unwanted libraries, APIs and DLL should be removed / disabled.
    c. User permissions should adhere to least privileges and a need to have model.
    a. Use a Defense in Depth strategy to include all the possible measures that need to be taken for all OSS, along with other products in the network.
11. **Patch Management:** The organization's Patch Management process should monitor and update patches released for the OSS.

a. Check the community associated with your open source code. These communities typically have bug-tracking systems and mailing lists that give information on known security issues and provide the latest news and exploits.
b. When a new vulnerability is identified, the organization should explore possible mitigation strategies that can be implemented until a patch is available.
c. Once a patch is released, test the patch for stability and applicability within your test environment prior deploying on your production systems.
d. Have a time bound approach to patch all vulnerable OSS in place.

12. **Compliance:** From time to time, verify the inventory of OSS, to ensure compliance with the OSS Usage Policy. Software composition analysis (SCA) products may be used to analyze application composition to detect components known to have security and/or functionality vulnerabilities or that require proper licensing.
13. **Business Continuity / Capacity Building:** Organizations should ensure that their employees have the relevant skills required to maintain an OSS. Trainings should be regularly imparted and skills distributed to avoid single point of failures. Knowledge bases of the OSS should be developed and maintained within the organization.

Appendix A – OSS Licensing

Open source licenses are generally classified as permissive, reciprocal (restrictive), or falling somewhere in between, depending on the nature of the restrictions and obligations contained in the license.

Permissive open source licenses: These are generally characterized by more lenient obligations, which may include allowing secondary distribution so long as the distributor provides attribution, or allowing secondary software alterations or improvements to remain proprietary. This flexibility facilitates the ability of end users or licensors to incorporate the licensed software into derivative products.

Example: The MIT license, the Apache Software License, and the Berkeley Software Distribution license (BSD)

Reciprocal or restrictive open source licenses: These are generally characterized by more stringent restrictions and obligations for end users or licensors of open source software. The distinguishing feature of a reciprocal open source license is the so called "copyleft" requirement. The copyleft requirement obligates those who incorporate and later distribute code to do so under same terms and conditions as the original piece of open source software, even if that software has been modified. Additionally, open source licenses containing copyleft obligations will frequently require licensors to reveal the underlying source code or make it available when requested by others. These copyleft obligations can create risky exposure for licensee's failing to monitor the terms and conditions governing the open source components being incorporated into such licensee's software code.

Example: The GPL