IMPACT OF CYBER SECURITY IN OPEN SOURCE SOFTWARES

Dr. Ramesh
Assistant Professor,
Kanya Mahavidyalaya, Kharkhoda.

Vinita Rani
Student of B.Ed. in Computer Science
Hindu College, Sonipat

ABSTRACT

It is often claimed that open source software is intrinsically more secure than closed source or proprietary software. Others argue that it is not, and it is expected this debate will continue for some time to come. The availability of source code provides both attackers and defenders opportunities to study code in detail and identify software vulnerabilities.

On the other hand, closed source software forces users to accept only the level of security diligence that the vendor chooses to provide. This paper discusses ways in which we can take advantage of the nature of open source software with regard to IT security. We also outline a number of best practices in open source software security that are recommended by the open source community, along with important points on using open source products safely within the organization.

Keyword: Software, Organization, Security, Open Source, Proprietary, Network, Community, Applications, Recommended.

INTRODUCTION

Open source software usually refers to software whose source code is “open” and available to anyone to study, use and adapt. According to the Open Source Initiative, the terms for the distribution of open source software must also comply with 10 criteria specified in the Open Source Definition.

Security of an information system depends upon its design and the components used for building it. Apart from the hardware, the major components i.e. brain of a computer or digital system is software. Therefore, how this software is written is a major deciding factor in determining the security of a digital system, be it a piece of code for some ROM, an operating system for a network device like a router or just an application like a web browser.

In 1996, the enquiry board, which reviewed the failed maiden flight of the Ariane 5 launcher of Euproean Space Agency, recommended that the definition of critical components should include software. The top 3 items out of the 10 criteria include:

1. software should be freely redistributable,
2. software must allow for distribution as source code as well as in a compiled form,
3. licences must allow modifications and for derivatives generated from the source code.

Lists of software licences that comply with the Open Source Definition are available at Opensource.org. Examples include the Apache Software License, the GNU General Public License (GPL), the IBM Public License, and the Microsoft Public License (Ms-PL).
The term freeware refers to software that can be used with no cost. Open source softwareis essentially freeware, but freeware software does not always make the source code available publicly.

TRENDS WITH OPEN SOURCE SOFTWARE

Open source software has been gaining in acceptance more recently, even in enterprise environments. In 2006, Unisys predicted that open source software would continue to gain acceptance from enterprise customers as a vehicle for deploying enterprise applications that are able to drive business growth and innovation at a lower cost per transaction.

In Europe, open source is considered a means of improving the competitiveness of theICT sector. In fact, as early as 2005, it was reported that nearly half of all European local government bodies were using open source software in some form.

While considering the role for home users, computers and digital systems like mobile phones, PDAs etc. have changed dramatically the way we live our lives. Most of the information that used to be only on paper or in hard files is now shared on the computers and the Internet. Be it the accounting information of its customers by bank, transactions history for online banking, examination results of its students by a university or college, sensitive records of police, armed forces, etc. almost everything finds its way to a computer file.

While computer and internet security might not be as important from the perspective of a home user, it is of tremendous value for small businesses, corporate and multinational companies, governments, military, etc. where millions and billions of rupees or dollars or even state secrets are at stake. Consider a bank's security system being compromised and money being transferred to some other account in some other bank. Or imagine a scence where a nuclear plants control system is taken over.

OPEN SOURCE VS COMMERCIAL SOFTWARE

One distinct difference between open source and commercial software is the availability of source code for review. Because the source code for open source software is publicly available, it can be used basically for free. Many organizations, in particular small- and Medium sized enterprises have chosen or are considering choosing open source software for economic reasons.

The free and open availability of source code is also considered to be an aid to software security because community-based peer review of source code can more rapidly help identify bugs or vulnerabilities in software. However, not everyone agrees with this argument.

Commercial software is mostly “closed source”. That is, the source code is not publiclyavailable. Because the source code is not available, there is a barrier against access to the code that attackers have to cross, resulting in less likelihood of vulnerabilities in the source code being exploited even though vulnerabilities do exist. Again, not all people agree on this. After all, an unreported or unidentified bug does not mean that a flaw will go away.
There is not yet any universal agreement on whether open source security is better than closed source security, or vice versa. Arguments on both sides are compelling[7] and it is expected that this debate will continue for some years.

OPEN SOURCE SOFTWARE AS A SECURITY TOOL

The best way to ensure that software is free of errors and vulnerabilities is to make a manual audit of the code. However, the process may be time consuming and long enough to be impracticle for projects involving length code. There are many automated tools available to scan a piece of code for any possible errors particularly those which are documented and quite common. Both proprietry and open source tools are available for this purpose. Some of these valuable open source tools are described below:

lint is one of the oldest tool which checks inconsisties and errors in the C code. A similar tool called nslint checks errors in DNS files and another tool weblint checks errors in HTML files. A similar source code scanner for C++ code is clint. Pscan and Cqual are similar tools that scan C source code for inconsistencies.

BOON is a tool that can find buffer overflow possibilities in C programs. MOPS finds vulnerabilities in C programs and checks whether a program conforms to paradigm of secure programming.

Flaw finder is a tool built using Python that can be used to audit C and C++ code.

RATS, the Rough Auditing Tool for Security is a source code scanner that can scan C, C++, Perl, PHP and Python source code.

The scanners should be periodically run on the source code during the development life cycle. The scanners will only highlight where the problem lies. Actual rectifaction of the problem still has to be done by the programmer.

A variety of security tools have been developed by the open source community. The most popular use of open source security tools in the industry can be categorized as follows:

1. Firewalls, such as iptables.
2. Intrusion Detection Systems, such as Snort.
3. Network Monitoring Tools, such as Multi Router Traffic Grapher (MRTG).
4. Security Assessment Tools, such as Nikto for web server scanners.

As there is no official support for these open source tools, the use of such software carries inherent risks. Special care should be exercised, and management approval should be obtained before they are deployed in the organization.

SOFTWARE SECURITY FOR OPEN SOURCE SYSTEMS

As discussed earlier, one characteristic of open source software is the public availability of source code, including potential criminals and attackers. Attackers are able to study source code and exploit vulnerabilities that may be due to programming flaws much more quickly. In addition, open source applications are usually developed jointly by volunteer contributions from groups and communities over the Internet. Attackers might also be able to contribute parts of the code to the software this way. Code level security usually depends on reviews conducted by those entrusted with maintaining the project or other contributors. However, it should be noted that closed source software could also suffer from similar problems if source code is leaked out to the public, such as the introduction of backdoors by disgruntled staff.

Efforts have been made by the open source community to improve software security and quality so as to mitigate vulnerabilities in applications and systems, including open source software. In general, open source software security is best achieved by following these best practices:

1. Maintain an inventory of all software being used, including open source software. The software inventory should also document the version, the hash value (such as MD5 or SHA-1) for verification of the integrity of the source code, as well as the website where the software was originally downloaded.

2. Check the availability of security updates and bug fixes for open source software regularly so that patch management processes can be followed regularly to minimize any loopholes in the selected open source software.

3. Change all default security settings in open source software as soon as it is installed. Configure the product in the most secure way possible by disabling unwanted services.

4. Test and scan the source code with code analyzers or auditing tools, such as

BOON (Buffer Overrun detection), Flaw Finder, RATS (Rough Auditing Tool for Security), and so on. Developers may also want to run compiler-integrated tools, such as Pro Police (or Stack-Smashing Protector, or SSP) from IBM which automatically inserts protection codes into source code that protect compiled programs.

5. Ensure that the open source application fully complies with existing network architecture if the application requires the opening of any firewall ports. This avoids any violation of the organization's firewall and security policy when a new application is introduced.

TO BE SECURE

All the things related to security described are serious issues and need to be addressed. To make our computer systems more secure we need to make the software that it runs, more reliable and free of errors and bugs.

SECURITY THROUGH OBSCURITY

First measure that a beginner and new software engineer would think of is to make the software code secret for no one to see. This is because; this is how traditional security measures are taken. To save your automobile from thieves, you lock it up in garage. Similarly, to secure your important papers you put them under lock and key in a safe or locker.

This paradigm of security is what is commonly known as *security through obscurity*. While such a scheme might work well in the above situations, it does not produce good results in the case of software.

OPEN SOURCE PHILOSOPHY OF SECURITY

Though it is a debate, it is now being widely accepted that instead of keeping the code secret, making it available to everyone for review and changes helps in making it more secure. While it might be hard to perceive this at first, consider the case of scientific research. Research produced by scientists is reviewed by peers for any errors and flaws, corrections are made and the same iterative process is followed before it is acknowledged

or is applied in the industry. Same is the case with security of the software. If the code is made secret as in the case of proprietary software, either it is not reviewed at all or possibly by only a handful of software engineers and programmers. Even then you do not have a guarantee of a back door being left intentionally. In fact, such an incident was reported in April 2000, when it was discovered that Microsoft programmers have inserted a back door in their FrontPage Web server software. The flaw was discovered four years after the release of software and this long period of time was due to the reason that the software code was proprietary and secret. Had it been open, there would have been no back door in the first place as a software code which is available for public review cannot have apparent provision for its own abuse.

A similar approach of openness has since long been applied to cryptographic algorithms. In cryptography, there is a maxim that security of an algorithm should not depend upon its secrecy. Therefore, famous encryption algorithms like RSA, SSL, etc. are not secrets. Everyone knows the algorithm. Only the keys are secret. Hence the test applied to cryptographic algorithms is:

1. Publish the algorithm and the source code.
2. Programmers are encouraged to find vulnerabilities and errors in the algorithm and code.
3. After the algorithm and code has been thoroughly reviewed and it has been shown that it cannot be compromised, only then it is approved.

Open source software goes under the same test as the above for cryptographic programs. The source code of the program is freely available to everyone to find errors and to fix them. As a result, it goes under a tighter scrutiny and as a result bugs and errors if there are some are purged in the process and the resulting code is more secure.

USE OPEN SOURCE PRODUCTS SAFELY IN THE ORGANISATION

To use open source products safely, organizations must consider the following:

1. Set up a well-documented security policy and ensure the policy is strictly adhered to. This policy should be revised, as business needs change.
2. Download open source products only from trusted sites, such as the official website of the software developer(s), to avoid potential risks from pre-inserted malicious code.
3. Download source code rather than a compiled package. In this way, source code can be verified against the MD5 / SHA-1 checksums provided, analyzed for security vulnerabilities and compiled for the organization's specific needs.
4. Study the product's documentation carefully for any explanation of the secure configuration parameters.

5. Check whether there is a reporting procedure should vulnerability be discovered in the product, and ensure all security issues around the product are well maintained and addressed.
6. Check regularly on common security vulnerability databases, such as CVE (Common Vulnerabilities and Exposures), for published information on any security vulnerabilities pertaining to the open source product(s) being used.
7. Adopt a "Defence-in-Depth" strategy so that various threats at various levels right from the open source product to the network can be fully addressed.
8. Provide appropriate training to in-house staff for the support and maintenance of open source products. Put together proper documentation for all the practices and configurations required in order to avoid problems that might arise due to job rotations or employment termination.

CONCLUSION

The adoption of open source software within an organization is not as simple as just downloading and running a free program from a website. There are a number of security concerns that should be studied, weighed up and determined before an organization takes the plunge into the open source world. In addition, both individuals and organizations need to keep in mind recommended best practices put out by the open source community. Organizations considering using open source solutions in the enterprise should be aware of all the points outlined in this paper.

REFERENCES

1. "Linux - the new target for threats?." http://www.gadget.co.za/pebble.asp?relid=362, 2005. Gadget.co.za.
2. M. Broersma, "Hack attacks on linux on the rise." http://news.com.com/2100-1001-943911.html?tag=cd mh, July 2002. News.com.
3. E. Hurley, "Scrap iis?." http://search390.techtarget.com/tip/1,289483,sid10 gci779497,00.html, November 2001. Tech Target.
4. S. Bekker, "Blaster worm exploits rpc dcom vulnerability." http://redmondmag.com/news/article.asp?EditorialsID=5912, August 2003.
5. "The default answer to every dialog box is 'cancel'." http://blogs.msdn.com/oldnewthing/archive/2003/09/01/54734.aspx, September 2003.
6. "Cert advisory ca-2000-04 love letter worm." http://www.cert.org/advisories/CA-2000-04.html, May 2000. CERT/CC.
7. S. Harris, "Cyber alert system debuts as super worm spreads." http://www.govexec.com/dailyfed/0104/012804h1.htm, January 2004. National Journal Group.
8. M. S. Mimoso, "Red hat brings se linux to fedora." http://searchopensource.techtarget.com/originalContent/0,289142,sid39 gci957636,00.html, March 2004. SearchOpenSource.com.

9. M. Russinovich, "Sony, rootkits and digital rights management gone too far." http://www.sysinternals.com/blog/2005/10/sony-rootkits-and-digital-rights.html, October 2005.
10. "Patching policy for x64-based systems." http://www.microsoft.com/whdc/driver/kernel/64bitPatching.mspx, September 2004. Microsoft WHDC.
11. D. Appleman, "Opening source: The ultimate customer security blanket." http://www.devx.com/opinion/Article/20513, March 2004.
12. "Creating a national framework for cybersecurity." http://www.acm.org/usacm/PDF/CRS cybersec.pdf, February 2005. acm.
13. S. Kuchinskas, "Microsoft's loss not linux gain." http://www.internetnews.com/ent-news/article.php/3313241, February 2004. InternetNews.
14. Auchard, Eric, "Phones, Car Engines Face Security Threats", Yhaoo News, as on 09-02-2005
15. Fleish, Brett D., "Grand Research Challenges in IT Security and Assurance", Proceedings of International Workshop On Frontiers of Information Technology, 2003.
16. Folkert, "Tools and Tips For Auditing Code", http://www.vanheusden.com/Linux/audit.html as on 13-08-2005.
17. Gaudin, Sharon, "India/Pakistan Virus Writers Take War Online" http://www.esecurityplanet.com/trends/print.php/2109 031 as on 17-05-2005
18. Ibrahim, Aazar and Hussain, Mukhtar, "Net-centric Organizations in Pakistan: Security and protection of IT infrastructure", Proceedings of International Workshop On Frontiers of Information Technology, 2003.
19. Jabeen, Farhana, "Internet and Pakistani Society", Proceedings of International Workshop On Frontiers of Information Technology, 2003.
20. D. A. Wheeler, "Secure programming for linux and unix howto." http://www.dwheeler.com/secure-programs/Secure-Programs-HOWTO/open-source-security.html, March 2003.