OPEN SOURCE SECURITY ANALYSIS

Evaluating security of Open Source Vs. Closed source operating systems

Carlos Serrão, Daniel Neves, Paulo Trezentos
UNIDE /ISCTE, Av. Forças Armadas, Edif. ISCTE, 1600-082 Lisbon, Portugal
Email: {Carlos.Serrao,Daniel.Neves,Paulo.Trezentos}@adetti.iscte.pt

Keywords: Open source, Security, Closed source, DRM, operating systems, Linux, enterprise systems security

Abstract: Open source software is becoming a major trend in the software industry. Operating systems (OS), Internet servers and several other software applications are available under this licensing conditions. This article assesses the security of open source technology, namely the Linux OS. Since a growing number of critical enterprise information systems are starting to use Linux OS, this evaluation could be helpful to them. To illustrate the fact that application security depends, above all, on the security of the OS underneath, we present the case of a DRM (Digital Rights Management) solution – MOSESOpenSDRM - implemented on top of the Linux OS, in the scope of the EU MOSES IST RTD programme. Some of conclusions hereby drawn are not compatible with some Microsoft funded studies that point to the fact that open source OS's are more insecure. This main idea was firstly present by the authors in the *Interactive Broadcasting Workshop - IST concertation meeting* hosted by the European Commission in September 2002 (Brussels).

1 INTRODUCTION

When assessing the security of enterprise systems, one key point should be taken into account: the security dependencies.
This is particularly true in desktop client applications where the security depends on the platform hardware security and on the OS where they run on.
The analysis exposed in this paper makes some considerations and tries to help answering a question so often discussed: "*Are open source operating systems more insecure than the closed source ones*?"
Security advantages and disadvantages of both systems should also be considered since the question might have different answers depending on the kind of application and existing threats.
In the example use case provided here, a DRM (Digital Rights Management) client and a server infra-structure, security is a major issue and its analysis will dictate if it is feasible or not to deploy on open source technology. In the same spirit, other applications are security dependent and need to have strong reasons to backup their choices concerning architecture and operating system decisions.

2 RELATED WORK

Many articles have been written about this theme. Some of them, favour the idea of "Security by Obscurity", and others point to "Security of many eyeballs", concerning the closed-source perspective in the former statement, and the open-source perspective in the latter one.
Closed source advocates the secrecy of source code as a critical security feature. It is based on the notion that secrecy is necessary to hinder intruders and, if a security exploit does occur, keep damage to a minimum (Anderson, 2001).
Others defend that the security benefits of open source software stem directly from its openness. Known as *the many eyeballs theory*, it explains that an operating system or application will be more secure when you can inspect the code, share it with experts and other members of your user community, identify potential problems and create fixes quickly (TrueSecure, 2001). Without the source code, we

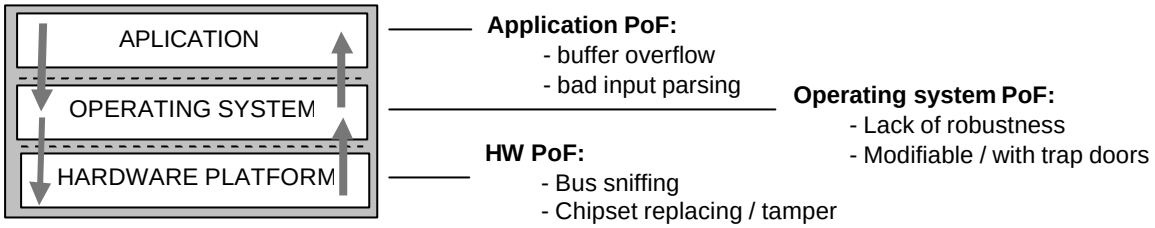

Figure 1: Architecture Points-of-Failure

remain dependent on your software vendor and on their strategic and economic agendas to correct any problems that occur.

The security issues stemming from closed source code were also addressed in a 1999 study, Analysis of the Security of Windows NT, by a group of Swedish researchers. The study's authors called attention to a number of problems including: weaknesses covering the entire Confidentiality / Integrity / Availability range; the sheer volume of bugs discovered; and reliance on security by obscurity.

According to the Attrition.org site, from August 1999 through November 2000, more than 56% of all successful attacks occurred on systems using Microsoft server software. Statistics kept by *alldas.de* show that, since April 2000, Windows operating systems suffered 63% of all successful Website defacements, compared to Linux-based systems (18%), Sun Solaris (3%) and other systems. While these statistics would seem plausible if there were three times as many Windows systems as GNU/Linux systems in use, that is simply not the case. These statistics provide strong counterpoint to the security by obscurity argument.

In short, closed source software is far from being a magic bullet in terms of security. The same secrecy that proponents claim safeguards to computing environment carries its own set of risks, along with a burden of hard and soft costs in terms of unanticipated downtime, lost productivity and threats to business-critical data (Aboul-Hosn, 2001).

The debate about the relative security benefits of open source and closed source code will continue for some time. “Even though open and closed systems are equally secure in an ideal world, the world is not ideal, and is often adversarial”. We can expect attackers to search for, find and exploit phenomena that break the symmetry between open and closed models.

3 OPEN SOURCE CONCEPTS

The growth of the open source software usage has lead to a misuse of terms and concepts such as “shareware” and “freeware” that don’t have anything to do with it.

“Free software” was the first concept introduced regarding the way applications were used and distributed. It was coined by Richard Stallman, head and founder of the GNU project and Free Software Foundation, back in the middle 80’s. The concept is legally supported by the GNU Public License (GPL) and some other compatible but more specific licenses.

In the beginning of the 90’s, Eric Raymond and Bruce Perens, among others, introduced the expression “open source” to designate not only GPL applications but a lot of other software under licenses that are not GPL compatible, but allow the user to access, change and distribute freely the source code (like BSD style). This is a less fundamental and “religious” perspective of the open source trend when comparing to Stallman’s ideas. The “open source” concept is widely explained at.OpenSource[1]. web site.

[1] http://www.opensource.org

Recognizing some of the open source business model advantages and pressured by legal actions, the software giant Microsoft allowed some of its partners to access some products source code. To this new licence Microsoft called "shared-source licence". This is having a very small impact since only a small group of people has access to the code and they still cannot change it accordingly to their needs.
In our paper and from this point forward, we will use the expression "open source" referring both to "free software" and "open source", but not including "shared source" applications in this group.

4 OPEN SOURCE OPERATING SYSTEMS EVALUATION

This section presents an evaluation of some security key points of an operating system. When reasonable, contrast between open source and closed source OS is pointed out.

4.1 Scope

The inherent (in)security of the PC platform is out of the scope of this paper. We assume that the risk of using this platform is taken by the software developer and the main concern is the operating system itself.
Some particular characteristics of operating systems are taken as features and not as threats. For instance, the possibility that one user process reads the memory space of any other process launched by the same user, could be a security concern, but it is a feature present in almost all OS. That feature is used by debugger applications, for example.

4.2 Points-of-failure

Security analysis should start by defining the potential points-of-failure (PoF) in the architecture. The image above (Figure 1) identities PoFs for each layer.
At the OS layer it is identified the lack of robustness and if it is modifiable and could have trap or back doors.
Each layer has security dependencies on the underneath layer.
The next sections will focus on the operating system's PoFs.

4.3 Security characteristics definition

The robustness and security of the operating system is directly related with five aspects: architecture strengths, its solid development, fast bug detection, improved bug reporting and effective issuance of patches.
Architecture strengths are a result if the structural options taken are appropriated for the environment where it will run.
Solid development is another important aspect and will be a consequence of the developer team knowledge, environment and organization.
The *fast bug detection* consists on how quickly flaws and errors are detected.
At the same time, the quality of *bug reporting* is also very important. How good are the bug reports submitted by users?
After the flaws detection, *effective issuance of patches* is crucial. How fast are bugs corrected?

4.4 Characteristics assessment

Each one of the characteristics pointed above is typically different between open source and closed source operating systems (Table 1).

	Open Source	**Closed Source**
Optimal architecture design	Medium	Medium
Solid development	Good	Good
Fast bugs detection	Good	Good
Bug reporting	Good	Poor
Effective issuance of patches	Good	Medium

Table 1 – Operating System's characterization

Most of the times, the *structural design* of operating systems is dictated by historical reasons and not by a careful evaluation of the best option.
Since Linux is into so many different types of markets (desktop, server and embedded devices) the overall structure is the one that best fits all of them: monolithic kernel with module loading at run-time. Nevertheless it can be optimized for specific applications, like an embedded multimedia device, through small modifications such as having a small footprint (Raymond, 1998).
On the other hand, closed-source software mainly has to assure compatibility with legacy systems and the optimal overall design is not taken into account (Anderson, 2001).
The *solid development* is a difficult matter to evaluate. Working teams in these different

environments act differently. The book "The Cathedral and the bazaar" from Eric Raymond (Raymond, 1998) argues that the Linux decentralized organization method is powerful even for huge software projects such as a Operating System development. In this environment, programmers are skilful, motivated and well organized.
But enterprise development teams are nowadays also very well prepared and with good tools for collaborative work. Closed source has in this sense also a solid development environment (Aboul-Hosn,2001).
In closed source operating systems bugs are *quickly detected*, since the number of users is usually large. Bugs are also quickly detected in open source because although the number of users is smaller, they are usually more active (TrueSecure, 2001).
Open source users generally provide better *bug reports* since they have a kind of "relation of trust" with the developers. In that sense, they fill more obliged to contribute. There is also another reason which is that users usually are more technical and therefore can provide a more accurate report of the OS flaw (TrueSecure, 2001).
Closed source OS users provide worst bug reporting since they have already paid for the product, so they don't feel the need of helping the developer team (TrueSecure, 2001). More, the users are, most of the time, less skilled in technical terms and the bug reporting is done in an automatic manner (Windows XP).
The quick issue of patches is the last characteristic analysed.
SecurityPortal[2] accounted that Red Hat (Linux Vendor) takes, in average, 11 days to respond with a patch to a discovered bug, in contrast with Microsoft that takes 16,1 days and Sun that takes 90 days (TrueSecure, 2001).
The reasons have to do with the fact that there are several different teams responsible for Linux services. A sole unique team is responsible for the kernel, but others are in charge of INETD, DNS, Webservers, editors, and other. So the bugs are therefore solved by different developers. Closed-source OS use their own software in the core services of the operating system and then suffer from the overhead of managing all the problems that occur.
There is other reason related with the fact that sometimes, in the open source environment, user submits the bug report and the bug correction at the same time.

4.5 Operating System threats

In the last section we pointed some characteristics that could compromise OS security and its robustness.
But there are some other threats such as the fact that the OS can be malicious if modified to have a different behavior then originally planned, thus making application security mechanisms fail on purpose.
This can occur only with open source systems like Linux and only if the user does not check the integrity of the code against a hash signature.
Taking the Linux example, this threat is not very realistic since it would require a good programmer to know where to change and introduce the malicious code in the kernel tree.
It is more or less easy to change kernel behaviour since kernel is freely available, and there are good tools and good documentation to support it. Nevertheless, it has a complicated structure so it has to be done by a skilful person.
But even if someone changes the OS kernel to compromise an enterprise information system, he has to deal with the problem of replicating and disseminating it.
In this case, the attacker has two options: (a) a source patch to kernel is disseminated to other users, but then the patch depends on the kernel versions and it is necessary to recompile all the kernel, or (b) the entire kernel is distributed but that would be a difficult task because it implies the distribution of a large number of files that are very related with the hardware components, which implies a later configuration. This is a complicated task to perform, specially the dissemination of such possible changes.
Closed source can raise other threats for the enterprise information systems. The fact that is proprietary and could be discontinued is important (TrueSecure, 2001).
For crucial applications there is also the problem that if the user does not have access to the source code, he cannot be sure that there aren't backdoors somewhere in the OS.

2 http://www.securityportal.com

5 SECURITY ANALYSIS OF THE DRM USECASE

This part presents the security analysis of a use-case that depends highly from the operating system - a Digital Rights Management (DRM) platform (both the client and the server parts).
Although the focus of this analysis is on a specific problem, this analysis can be easily extended to a wide number of enterprise information systems and industry technologies.
DRM is the chain of hardware and software services and technologies ruling the authorized use of digital content and managing any consequences of that use, throughout the entire life cycle of the content (Duhl, 2001).
The Internet's global reach enables the transmission of content on an unprecedented scale. At the same time, cheap, abundant digital technology makes it possible to create and efficiently distribute all forms of content and information in digital format, including text, images, email, audio, video, software, and games. Widespread Internet penetration, combined with digital technologies, permits anyone with these tools at their disposal to easily make a potentially unlimited number of copies of a piece of content without any degradation in quality and to distribute a single copy to an unlimited number of users. As a result, unprotected digital content is extremely vulnerable to theft, unauthorized access, and propagation (Ebrahimi, 2001; Conan, 2002).
DRM technology has been developed to protect the commerce, intellectual property ownership and privacy rights of digital content creators and owners as it travels through the chain, from producer to distributor to consumer and, even farther, from consumer to other consumers (by consumer, we mean any recipient of the content). It persistently protects and governs content based on usage rules specified by the content owner and rights held by the consumer. DRM can be used to control and track authorized access and use for marketing, sales, and royalty, penetration, and accountability reasons. For these reasons, DRM can be an important component of an organization's business strategy (Conan, 2002).
Different types of organizations may have different motives for protecting and managing their digital content. Content owners and service providers may want to control access to their content in order to generate revenue from its sale, while an enterprise may want to share content but not sell it. In an enterprise, where content is shared but not sold, access to content is generally controlled through username/password authentication. This, however, does not control the policy or what users can do with the content once they have access to it. DRM provides three benefits: 1) persistent protection of content through encryption, 2) expression and association of usage rules with content, and 3) enforcement of the usage rules (Lampim, 2001).

5.1 Architecture

The MOSESOpenSDRM architecture (Figure. 1) is adaptive (Serrão, 2002), which means that it can be configure for use with several business models and different types of content. MOSESOpenSDRM deploys a traditional DRM solution for content rights protection and can be applied for publishing and trading of space and planetary imagery. Additionally, the security architecture proposed is inline with the recent international specifications OPIMA, MPEG-4 and MPEG-21 as well with some of the proposals for JPEG2000 standard Part 8 – JPSEC – JPEG2000 security (Serrão, 2002).

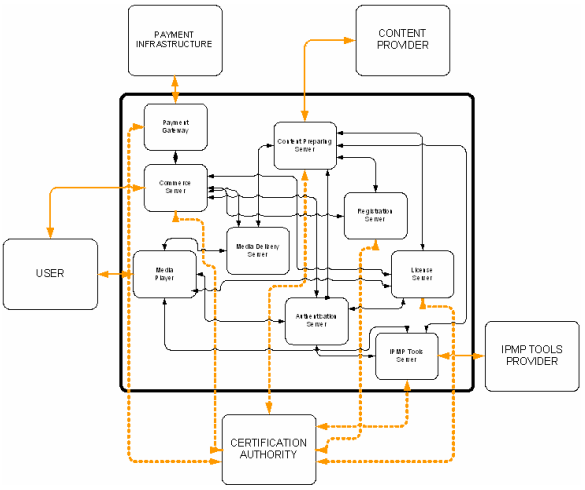

Figure 2 : MOSESOpenSDRM detailed architecture

This DRM solution is composed of several optional elements covering the content distribution value chain, from content production (content author or producer) to content usage (final user). It covers several major aspects of the content distribution and trading: content production and preparation (Content Preparation Server, Registration Server), content protection (Registration Server, License Server, Intellectual Property Management and Protection - IPMP tools server and Authentication Server), content interactive distribution (Media Delivery Server), content negotiation and acquisition (Commerce Server, Payment Gateway), strong actors and users authentication (Authentication

Server) and conditional visualization (Media Player, IPMP tools Server, License Server) (Serrão, 2002).

This architecture provides an integrated DRM solution, interfacing with several external actors which have their own specific role and requirements: User (requires access to content), Content Provider (wants to make content available for trading, be assured that this content is protected and that it receives a financial return for the traded content), IPMP tools Provider (wishes to commercialise their own content security tools), Payment Infrastructure (represents the financial environment) and the Certification Authority (the entity responsible for injecting recognised trust on the system).
As mentioned MOSESOpenSDRM, currently being developed at Adetti, is being applied to several R&D projects and has been submitted as reference architecture for the JPEG 2000 security extension, JPSEC.

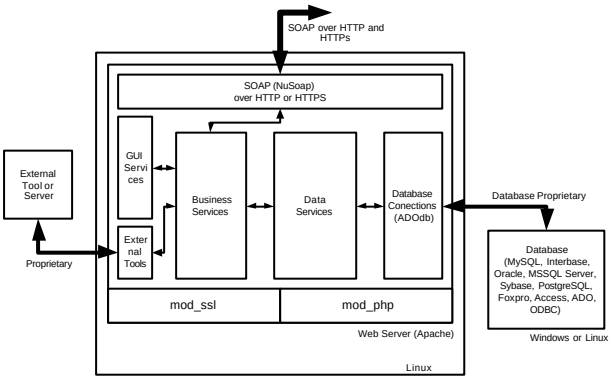

Figure 3 :. MOSESOpenSDRM components architecture

Each of the components in the MOSESOpenSDRM server components has a similar architecture. All of these components are based on the new distributing computing called Web Services and therefore exchange information using SOAP over HTTP or HTTPS (HTTP for non sensitive information and HTTP for sensitive and secure information). All these components are entirely based on OpenSource technology: PHP programming language, Apache Web server module, mod_ssl and OpenSSL, mod_php, PHP NuSoap, PHP ADOdb and Linux. Basically the security of these components relies both on the socket communication layer (through the usage of OpenSSL and mod_SSL to provide SSL/TLS secure communication channels) and on the application layer (developed using the PHP programming language).

5.2 Weighting security risks

It is common sense that authors of digital multimedia content should and must be paid by their work. The problem is how to make that possible, when there is an installed sense of multimedia content freedom (a particular example is the MP3 case) (Guterman, 2001).
Two different scenarios can be considered. Firstly, not charge any fees when user access or copy the content or, secondly, the user can only access the content if he has permissions and have paid for it (Fisher, 2001).
In the first case, the trust in independent organizations is established (like national authors association) for distributing the profits between authors accordingly to a "play list chart" or any other ranking mechanism. The profits to be shared come from a tax over the medium selling transaction, like selling blank CDs (Haes, 2001).
An alternative to this scenario could be accomplished with a charging mechanism when user could voluntarily choose to pay if he likes the content. In this case, content is always in clear.
In the second scenario, the user can only access the content if he has permissions. Permission is granted based on a previous commercial transaction.
A DRM architecture should be established to deploy this scenario involving content providers, access providers, financial entities and users. This is the approach being followed by the MPEG IPMP-Extensions group.
It is possible to identify several problems of the first approach: a) the charge through a tax implies that is legally possible to impose such tax. The heterogeneous set of laws of the different countries around the World and at the same time, globalization, makes it very hard to achieve it in a synchronized way. Industry and governments should prevent the settle of a parallel economy that would provide the same medium (e.g. blank CDs) without the tax and therefore with a lower cost; b) if the charge were made through a voluntary contribution we are assuming that at least a major part of the users are honest and that is possible to have a global infrastructure that allows users to choose/vote/poll when they like the content and they are willingly to pay for it. There are not many examples of economic activities that assume that users are completely honest. People might even be honest with non-profitable organizations but will they be honest and pay to multinational corporations?
Moreover, the infrastructure that is needed for the poll/pay for content assumes that the user is on line or will be in the future. Technically, to deploy such a

solution in a wide environment could be as difficult as implementing a global DRM solution.

Let's now analyse the MOSESOpenSDRM architecture and the open source operating system security implications.

The architecture runs over two insecure platforms: PC hardware and desktop Operating Systems (Linux or Windows). Therefore is not perfectly secure.
We can evaluate the security risk of deployment a solution with a ratio like this one:

$$\text{Risk} = \frac{\text{Risk of someone breaks it} * \text{Consequences of that break}}{\text{Global profit of having the system}}$$

The product of the "Risk of someone breaks it" by the "Consequences of that break" will give us an idea of the potential losses.
As much as the upper part decreases and the lower part increases, the better for the devised solution.

RISK OF SOMEONE BREAKS MOSES OpenSDRM APPLICATION -
The risk is fairly high comparing to the use of a Set Top Box, but is strong enough to avoid that the large majority of hackers/crackers could break it.
Breaking the system means that the violator can gain access to some specific content without any payment. It implies that IPMP had been cracked or the cryptographic keys have been compromised. The cracking is somehow "limited" by:

i) it just applies to content "managed" by that IPMP tool

ii) IPMP could be replaced on-the-fly (since it is software)

ECONOMICAL CONSEQUENCES OF THE BREAK -
The consequences are limited. As we said before, breaking the IPMP tool only allows one to access content protected by that IPMP tool but all other contents are still protected. This is clearly different of the DVD case.
If the IPMP tool broken becomes widely use, we can then issue a new IPMP tool with the flaw corrected.
However if a new IPMP tool is issued does the user loose the possibility to access to the older content? No, because it is possible to have two IPMP systems loaded and the content is the one that defines which IPMP is needed for the payload decryption.

GLOBAL PROFIT OF HAVING THE SYSTEM -
The global profit of a DRM or of an Enterprise Information System change a lot and its accounting is out of the scope of this paper.

Weighting the 3 points, MOSES DRM minimizes the first 2 and therefore allows the growth of the third one.

The use of the open source software under the DRM layer is possible without compromising the security. As we present in the previous sections (sections 5.4 and 5.5) open source operating systems are strong candidates both for desktop and server systems.

6 CONCLUSIONS

Some recent studies argue in favour of closed source operating systems. The main reason presented is that since the code is not available there are fewer chances to discover hidden and unknown bugs of the system.
This false sense of “security by obscurity” is nowadays becoming less popular but is still used.
Nineteen century Kerckhoff's principle that states that we should assume that enemy know one's cipher system and so security should only reside on the key, can easily be transposed to nowadays Operating System debate. With reverse engineering tools and techniques at our disposal OS flaws can be easily detected and explored even in closed source systems.
On the other hand, in a closed source perspective it is really much more difficult to change the operating system, although this is not an advantage if we easily manage to break it.
The DVD / DeCSS case is the best example how closed source doesn't necessarily mean security.

Although during this paper an independent evaluation of operating system characteristics have been proposed, proving that all open source systems are secure was not the purpose of the authors.
Nevertheless, they hopefully stated good reasons to one consider that there are not any technical evidences that “closed source systems are more secure then open source ones”.

ACKNOWLEDGEMENTS

The authors would like to thank ADETTI (Associação para o Desenvolvimento das Telecomunicações e Técnicas de Informática) by their support to this work, and the means that were put at our disposal.

REFERENCES

Raymond, Eric S. 1998, «The Cathedral and the Bazaar», at http://tuxedo.org/ ~esr/writings/cathedral-bazaar/

Anderson,Ross «Security in Open versus Closed Systems - The Dance of Boltzmann, Coase and Moore»

TruSecure Corporation, 2001 «Open Source Security - White Paper»

École Polytechnique Fédérale de Lausane, 2002, «MOSES security overview»

Aboul-Hosn, Kamal and Collinge, Matthew and Edwards, Chris 2001, «Comparison: Open Source Software Vs. Closed Source Software»

http://www.opensource.org/advocacy/case_for_business.html, «Open Source Case for Business», 2002

Lampim, I., "D22 - User's requirements for remote sensing applications", IST Project 28646, PRIAM, 2001

Ebrahimi, T., "JPSEC Scope and Requirements 1.0", ISO/IEC JTC 1/SC 29/WG1 N2388, 2001

Conan, V., Rollin, C., "JPSEC Scope and Requirements 2.0", ISO/IEC JTC 1/SC 29/WG1 N2548, 2002

Duhl, J., Kevorkian, S., "Understanding DRM systems - a IDC whitepaper", IDC/Intertrust, 2001

Serrão, C., Conan, V., Sadourny, Y., "JPSEC – Protecting the JPEG2000 code-stream", ISO/IEC JTC 1/SC 29/WG1 N2650, 2002

Guterman, J., "Click and Play: Your Guide to Music Sites on the Web", Business2, http://www.business2.com, 2001

Fisher, W., "Digital Music: Problems and Possibilities", http://www.law.harvard.edu, 2001

Merck A. "MP3.com - business model and development", MEBIS, 2000

Haes, J., Hummel, J., "Napster and the Economics of Music Distribution", NetAcademy, http://www.netacademy.org, 2001