Perspectives on the Security of Open Source Software

Man Xiong, Lin Huang, Ahmed Tolba, Winfred Wong, Eric Vandenberg, Mohammed El-Gammal

December 9, 2004

Abstract

The paper examines the security of open source software from three different perspectives. First, we investigate how development processes affect security. Several open source software development models are described, and the meaning of openness and importance of testing are analyzed. Next, we move on to explore the economic impact and interests of security. Monetary value of security and the incentives behind various open source communities are discussed. Finally, we take a look at the effects of government policy on open source security. We describe the policies in the United States, Germany and China, and examine how different roles of the government impact security of open source software.

Table of Contents

1 Aspects of the Development Process and Impact on Security

1.1 Introduction

The strongest arguments about the security or vulnerability of open source versus proprietary software revolve around the very making of it. Aspects of the development process have a great impact on how solid software is. The section goes through the open source development process citing three of the most prominent open source projects. It then moves to detailing some of the published data on the security of open source. Finally it takes a deeper look at two of the most influential features of the development process related to the security of open source, which are openness and lack of formalized testing.

1.2 Open source development process

Open Source Software (OSS) is different from traditional software produced by companies like Microsoft, Apple, Sun, etc. In commercial environments, companies have the monetary resources to hire programmers, managers, system architects, and the ability to set and direct large project schedules. With open source most contributors are volunteers. They tend to join a project or two, before finding some other new activity that captures their interest. They like to work on what they find fascinating and as a result it is difficult to direct and co-ordinate their collective efforts. Given family, job, and social commitments it is very difficult to reliably create project schedules. In order for open source projects to be successful, they need a framework for development that can handle a high turnover of contributors. In the following sub-section we take a look at examples of successful open source projects and how they are organized.

1.2.1 Linux Process[1][2][3]

Linux is an open source Operating System (OS). Its development work is broken into two branches, the experimental branch and the production branch. All new features are added to the experimental branch for initial testing. The experimental code may be unreliable resulting in data loss, machine failures, etc. The production branch contains only well defined and tested features, it is much more reliable and has fewer releases. Companies bundle production versions of the Linux core OS with other packages to provide a generally usable Linux distribution.

---

[1] Andrew Morton, *Keeping the Kernel - Insight and wisdom from the maintainer of the Linux 2.6 kernel*, Linux Magazine, Aug 2004

[2] The Linux kernel mailing list FAQ – kernel.org

[3] Open source development labs - osdl.org

The Linux core subsystem has over one hundred individual maintainers. A subsystem maintainer takes primary responsibility for that component. This includes monitoring bug reports, arranging for fixes, coordinating, approving and reviewing all code changes. In addition they answer inquires and work with other core maintainers to ensure the quality of the production branch. Maintainers are selected by Linus Torvalds, the founder of the Linux project. They are typically active and passionate developers in the area they are assigned.

The features that go into Linux release are decided through an iterative process of discussions, reviews and negotiations between key core OS developers and companies releasing Linux distributions. Once a feature set has been defined, agreed and published; the work begins in the experimental branch. As a Linux release date approaches, Linus announces a feature freeze, which means no new features will be accepted until the next stable release. Only critical fixes are allowed. This is a short phase usually preceeding the creation of a new stable Linux branch.

The process of getting features or patches into Linux core OS is straightforward. To get a change into the build, a developer sends an email to the component or subsystem maintainer. The patch must be a self contained logical unit with no dependencies on other patches. The maintainer's job is to review the changes and ensure other developers working in that component review it. The patch is well tested before prior to inclusion in the production branch.

1.2.2 BSD's Process[4]

The development model of the three BSD variants, FreeBSD, NetBSD and OpenBSD, was originally created by the Berkeley's Computer Systems Research Group. For the purpose of this section we will focus on FreeBSD. The entire FreeBSD project, including all source code, documentation, bug reports, mailing-list archives, and even administrative data is maintained in a publicly readable source-code-control system. Anyone may review the source code and existing bug reports, track progress on fixing bugs, and post bug reports. Anyone may join or participate in numerous FreeBSD mailing lists. There are three groups of people that directly work on FreeBSD, they are developers, committers, and the core team.

There are three to four thousand developers, each working on some part of the system like the FreeBSD core, system utilities, documentation and updating other OSS projects in the FreeBSD port collection. Similar to the Linux development process, developers have read access to the source code repository but can't change it directly. They work with a committer to get the changes into the source code repository.

---

[4] Marshal Kirk McKusick and George V. Neville-Neil, *The design and implementation of FreeBSD operating system*, Addison-Wesley, 2005

There are three to four hundred committers. Most committers specialize in some part of the system and are not allowed to change other areas. All nontrivial changes should be reviewed by one or more committers before being checked into the source tree. In addition to reviewing and committing the work of others, they are often developers themselves.

Committers nominate other developers for advancement into becoming a committer. Often this happens when they have worked together well. The core team runs decides on approval based on previous work evaluation and initial scope of new work.

At the center of the project is the core team. The core team is composed of nine people, elected every two years. The committers elect the core team from their members. The core team acts as the final gate keeper of the source code. It monitors what is being committed and resolves conflicts if two or more committers can not agree on how to solve an issue. The core team approves developers' promotion to the status of committer. They also can temporarily or permanently demote someone back. Like Linux, FreeBSD has stable and current unstable releases.

1.2.3 Apache Process[5]

Apache is the most popular web server on the internet. It was started by a group of developers to fix known problems with The National Center for Supercomputing Applications (NCSA) web server. Since then it has evolved into a full fledged open source project with an informal organization responsible for guiding its development effort, the Apache Group (AG),

The AG in its entirety consists of volunteers working part-time on the project. None of the developers can devote large blocks of time to the project in a consistent or planned way, which is typical of open source. The decision making process is based on a decentralized model with asynchronous communication.

AG members are people who have contributed for an extended period of time, at least six months, and are nominated and voted for membership by existing members. AG started with eight founding members, and went up to twelve for most of the period covered by the cited study. Now there are twenty-five members of the core development team.

Each AG member is expected to use their own judgment about committing code. There is no rule prohibiting any AG member from committing code to any part of the server. Votes are generally reserved for major changes that would interfere with the work of other developers. Although there is no single development process, each developer

---

[5] A. Mockus et al, ACM Transactions on Software Engineering and Methodology, Vol. 11, No. 3, July 2002

iterates through a common set of actions while working on the project. This applies whether a new problem is discovered or new functionality is identified, determining whether a volunteer will work on the issue, identifying a solution. It also including developing and testing, presenting changes to the AG for review and committing the code and documentation. Depending on the scope of the change, many iterations may be required; it is generally preferred that all the changes by taken together though.

1.2.4 Characteristics of OSS development Process

Although there are differences between the development processes of open source projects, generally they have a lot in common. For instance,

- **Volunteer developers** - The vast majority of open source developers are volunteers. A study done on open source projects found that 73% of open source projects have only one stable developer, and less than 5% has 20 or more stable developers[6]. Because open source developers are transient volunteers, they tend to work in areas they find interesting.

- **Open source code** - Source code is available to anyone to read and critique. In addition, anyone can submit changes for inclusion in the source code. Typically there are a few designated developers who approve code submissions.

- **Core developers** - As mentioned above only 5% of open source projects have a stable set of developers of 20 or more. As the number of developers the need for a more formal structure increases, which leads to formation of "core teams" of developers. Core developers mostly come from the set of stable developers and play the key role in setting the future direction of a project; they are the most influential stakeholder.

- **Developers are users** - Open source developers are also users of the software. Active developers in an open source project install the latest releases of the software which they use in their day to day work. This means that open source developers are a subset of the open source user community.

- **Short Release Cycle** - In the active or experimental branch there are frequent releases and developers are expected to self-host the most recent release and report problems as they find them. This is surprising given the part-time nature of OSS developers.

- **Lack of formalized tests –** OSS developers generally write feature/unit tests, and conduct peer reviews in an areas. Other than that, open source contributors tend to be

---

[6] Capiluppi and et al., *Characteristics of Open Source Projects*, Proceedings of the Seventh European Conference On Software Maintenance And Reengineering (CSMR'03), IEEE computer society, 2003

more interested in coding than documentation or testing. Most open source projects don't have formal integration and system-wide tests that are run before interim releases. They mostly realy on the community at large to do the testing indirectly through using the software.[7].

- **Source Availability (Openness)** – Open source projects have their source code available for public viewing and modification generally under one of two licenses which are the GNU Public License and the BSD style license.

The next section discusses last two points as the basis for an argument about the impact of development process characteristics on the security of open source. These points are at the center of the discussion of the inherent security of open source and proprietary software. First we should look at the data to give the reader more insight into some of the relevant security statistics that were published recently.

1.3 Development Characteristics and Security

1.3.1 Openness

The Alleged 'Power of Openness'

At the very heart of the open source movement lies the predicate of software 'freedom'. This freedom advocates for source code to be freely available for viewing and modification. Bollier describes the process producing this freely available source code by means of passionate developers working for the betterment of society through 'gift economics'. According to him those developers complete the cycle bringing software back to its proper place where it started, which is the public pool of society where its benefits can be shared[8].

Among the most stated reasons for the claimed security of OSS over its proprietary counterpart is the availability of source code. Eric Raymond's claim that "given enough eyeballs all bugs are shallow" represents one of the most fundamental pillars of the belief among open source advocates. In the cathedral and bazaar[12], the argument goes like:

> *"In the cathedral builder view of programming, bugs and development problems are tricky, insidious, deep phenomena. It takes months of scrutiny by a dedicated few to develop confidence that you've winkled*

---

[7] Cristina Gack and Budi Arief, "*The Many Meanings of Open Source*", IEEE software, Jan 2004

[8] Bollier, David. *The Power of Openness*, April 1999.

them all out. Thus the long release intervals, and the inevitable disappointment when long-awaited releases are not perfect.

In the bazaar view, on the other hand, you assume that bugs are generally shallow phenomena—or, at least, that they turn shallow pretty quickly when exposed to a thousand eager co-developers pounding on every single new release.

Accordingly you release often in order to get more corrections, and as a beneficial side effect you have less to lose if an occasional botch gets out the door."

Whether openness of source code could in itself be a reason for security is the focus of this section.

All Eyeballs are 'NOT' Created Equal

Like other classes of flaws in software, security vulnerabilities could be viewed as mistakes made inadvertently by software developers either caused by lack of attention or lack of knowledge. Secure Socket Layer (SSL: A utility library that provides secure connections over the internet) is a good example. SSL is a frequently the source of issures as developers due to the lack the awareness of its proper use.[18]. Also, it's worth noting that people are sometimes too arrogant or are completely ignorant of their limitations. This could explain the observation of security expert John Viega. He said many developers are lousy security coders and reviewers even though they might not know it. Given this, the above argument implies a lot of reviewers may be looking at a product's source code but very few count. About the only bugs found are the easy ones, like the infamous 'buffer overrun'.

Another problem is the limited capacity of the human mind and its inability to comprehend all the interactions among complex pieces of software. In our opinion, complexity is not only the main source of bugs but also another impediment to the security review process. This problem may be compounded by the reviewer being unfamiliar with the software. Facts like this would render the casual review, and even the more serious one without the right expertise, all but effective. Reviewers with the right security expertise in the problem domain are not easy to come by. Scarcity implies an increased price for the same demand and this higher price makes it less likely that they would be reviewing open source Software18.

Finally, it's only fair to point the reader to the fact that exceptions to the rule do exist. Corporate backers like IBM could conceivably hire experts to derive security in open source products of interest to them. Also products like Apache are reputed for having a good security record and their onboard security expert. The ability of open source to get paid time from a qualified security expert is not the point being debated. The point is that even if OSS could score some points in this arena, this puts commercial software at least at the same position when it comes to getting expert security attention.

Now, Who is 'Reaaally' Looking?

It is plausible hackers (real hackers and not open source developers) could specialize in certain types of vulnerabilities before getting really good at finding them. They too have unrestrained access to the source code which makes their task much easier. It is not too hard to assume that even for the most skillful of them providing easy source code access reduces their operational overhead. They no longer need to use the crafty tools to disassemble and analyze software in binary form which would have been the case previously. They can become intimately familiar with the area they want to exploit. The argument made earlier in the section about the bar for security reviewers being high seems contradictory to the statement about hackers finding holes in source code. This can be clarified by pointing out the goal of the security review is to get rid of exploitable vulnerabilities while all a hacker needs to do is find one exploitable hole.

From another perspective, we can consider the Forrester's study claim that "Real vulnerability to attack begins with disclosure"[9]. One of the principal methodologies hackers employ is to look at an issued fix and compare it to the original module in an attempt to discern the vulnerability lurking in it. Next they use the window of time (and opportunity) before people apply the fix to exploit their machines. Considering the fact that reverse engineering and comparing modules is an order of magnitude harder when you don't have source code, having sources allows for a speedier understanding of the vulnerability. Speed is of the essence since the window of time is closing in before potential victims apply the patch. Finally, there is another factor outside the context of this discussion that is mentioned for its relevance. For Linux distributions a patch that is independently issued for a bundled component could be held till the next distribution release cycle. This could give the hacker plenty of time to react. This is particularly true in Linux distributions that do not provide and out of band auto-update mechanisms. Distributions that provide an out of band update include prominent ones like RedHat and Debian.

You need to leave your door unlocked!

Ignoring social and religious arguments (which we haven't got into due to the lack of relevance to this study), in our minds, 'the subsection title' is exactly what openness in software amounts to when it comes to security. Given that it is easy for a skilled felon to pick a lock or an experienced bank robber to demolish the vault door or really persuade the clerk under gunpoint to open it, then one could argue that those practices are not worth their while. If such argument was adopted then what this would translate to is slightly lowering the bar for less skilled bandits and making it easy for the professional ones. For software, we are talking about an area where how fast an exploit is produced makes a difference, so even deterring the hackers for a few days trying to works their way in the dungeons of the binary could be of help. 'Security through obscurity' is bad either and is not what the argument is designed to support. In this

---

[9] Koetzle, Laura, *Is Linux More Secure Than Windows*, 19 March 2004

regard, this would be like having a lock that doesn't lock or a vault door that opens with random unauthentic secret keys. What we fully support is well designed, 'expert' reviewed systems that are even more protected by allowing only relevant stake holders to view their source code, for extra assurance and independent reviewing.

The Utility of Tools and Utilities

Given the previously discussed argument of the complexity of large software and the inability of the human mind to fully comprehend all dynamics tools seem like a more promising avenue. Unlike the human mind, machines can handle systematic complex analyses flawlessly. Currently all major successful tools are at the early stages of research and development. Commercial companies have the advantage over open source given the abundance of resources and the centralized organization available for investment in complex code analysis tools. There is nothing stopping the open source community from developing such tools by leveraging the wider community of volunteers and academic researchers. An obvious example is Engler's work in Stanford[17]. Note that all tools developed in the open source are also available to enhance the quality and indirectly security of proprietary software.

1.3.2 Testing

Software testing is one of the most fundamental assurances for the high quality of a developed product. Quality of software represents consumer satisfaction across the breadth of a products´ features, including assurances about safety, privacy and security. The commercial software industry typically employs Quality Assurance (QA) technicians through a dedicated QA department. The area of formal testing is identified as a major difference between the commercial and open source projects. This section examines the difference in testing methodologies in open source and commercial software, its effect on the development process and finally the impact on software security.

System Wide and Product Integration Testing

With the exception of non-representative projects like Mozilla, open source could generally be characterized by a lack of formal system-level and integration tests. Mozilla could probably trace its half dozen test teams to inherited processes from the commercial days of Netscape. It is widely believed that responsible development practices like unit and feature tests to verify the correctness could be in use on both sides of the free and commercial software divide. Nevertheless, unit and feature tests are not deemed sufficient on their own in the mainstream software industry. System-level tests are equally important if not more crucial to verify new functionality doesn't break existing code. Modifying complex software is in a sense like manipulating the human body. Even though it might not be obvious at first that operating on one organ can have its effect on a totally different one, it's certainly likely given the complex processes that within the human body.

Impact of Lacking System Integration Tests

The below numbers are taken from a case study on Apache, a prominent open source web server by Audris Mockus et al. in 2000. It provides comparative defect numbers for Apache and similar undisclosed commercial products from the telecommunications industry. The analysis depends on a strict definition of defect. A defect, as far as the study is concerned, is a result of a Problem Report (PR) by customers or internal test. Defects are measured per 1000 of lines of code added (KLOCA) and 1000 deltas (KDelta). The stated reasons for such normalization (scaling defects by means of KLOCA and KDelta) are not to give an advantage to larger code projects. This would be bad from a software engineering point of view, as it would dilute the defect density by not taking into account the incremental nature of products[10].

Table III. Comparison of Defect Density Measures

Measure	Apache	A	C	D	E
Postrelease Defects/KLOCA	2.64	0.11	0.1	0.7	0.1
Postrelease Defects/KDelta	40.8	4.3	14	28	10
Postfeature test Defects/KLOCA	2.64	*	5.7	6.0	6.9
Postfeature test Defects/KDelta	40.8	*	164	196	256

The first observation is that post-feature test and post-release defects of Apache are the same due to the lack of system-wide integration testing. Regarding the post-release numbers, Apache ranks worst in terms of consumer defects. This clearly asserts the importance of system integration tests and the impact of them in the commercial world.

An Interesting Side Effect

Another observation we like to make note of is that post-feature numbers seem to be better for Apache than its commercial counterparts. This can shed some light on the software development process and diligence of developers in open source knowing that they have to ship high quality the first time and knowing that there is no QA team to back them up. Apache is known to be a high a quality product especially when it comes to security. In order to make sure that this observation is not unique to Apache, post-feature test numbers from the same study for Mozilla are also compared. Looking at the table below the low defect numbers seem to hold for Mozilla too, making this an interesting aspect of the quality of open source Software and hence its security[11].

---

[11] Mockus, Audris et al, *Two Case Studies of Open Source Software Development: Apache and Mozilla*, July 2002.

Table VII. Comparison of Post-Feature-Test Defect Density Measures

Module	#PR/KDelta	#PR/KLOC Added
Apache	40.8	2.6
C	164	5.7
D	196	6.0
E	256	6.9
/layout	51	2.8
/js	19	0.7
/rdf	27	1.4
/netwerk	42	3.1
/editor	44	2.5
/intl	20	1.6
/xpinstall	56	4.0

It's a well know fact that QA is not a substitute for a lousy development process. It is typically there as a quality gate ensuring the product is sound before wide spread deployment, catching few, if any, non-fatal bugs. If the lack of system testing would derive the development process to be sound in the first place then this is an interesting side effect that might balance the picture. If this is true it could mean that the bulk of open source defects are non-core issues. Given the speed of response to defects that is exhibited by some open source software, it could in some areas be a viable alternative to commercial software. Apache web server is a clear example of this.

Post-Feature Test Bug Rates in OS vs. IP

There could be another side effect happening across board, which could provide supplemental explanation for the high rate of post-feature test bugs in the commercial world. Working for a company where system test is part of the process (especially when this company is a large corporation) developers tend to be over reliant on the fact of system test backing them up. In some cases the QA organization is reduced to be a guinea pig of the developer. In such an environment, unit tests tend to slim down and sometimes even totally disappear. Whole features are committed to the source tree with no significant developer verification since it's someone else's job to make sure it 'works'.

System Test is Good

The section is by no means arguing against system wide tests but is pointing out the interesting side effects that could result from abusing the system on the commercial side and the extra diligence for the lack of it on the free side. We believe that if QA abuse is true on the commercial side then abiding by good development practices like unit tests and developer diligence while reaping the benefit and the extra assurance of system testing could boost the quality and stress the competitive edge that it has in this area. Consumers can reap the benefits of all of this by having a super reliable system upon delivery that could be deployed with more confidence.

Open Source Security Data

There have been several claims by the open source movement that, because of the characteristics of open source projects – open source, volunteer developers and many eyeballs looking at the source[12], then open source software is inherently more secure than IP based software. In this section we will look at arguments for open source inherent security and data collected by several sources that track security advisories issued by both open source and IP based software.

1.4 Security Statistics

As mentioned above there have been several claims by the open source movement that, because of the characteristics of open source projects, that open source software is inherently more secure than IP based software. In this section we will look at data and correlate it to the analysis from the previous section.

1.4.1 Security of open-source software vs. IP-based – Sources

Along with IP-based software vendors like Microsoft and Sun; and open source software distribution companies like RedHat and Novell; there are several services that track and publish advisories for vulnerabilities in software products. Some of these services are non-profit, university or governmental services. Table 1 lists some of these services. We will be relying on data from different services and software vendors' security advisories.

Source	**Organization type**
Cert	Carnegie-Mellon
Microsoft	Vendor
Red Hat	Vendor
Secunia	Service
SecurityFocus	Service
US-Cert.	Government
Zone-H	Service

Table 1. Source for vulnerability statistics

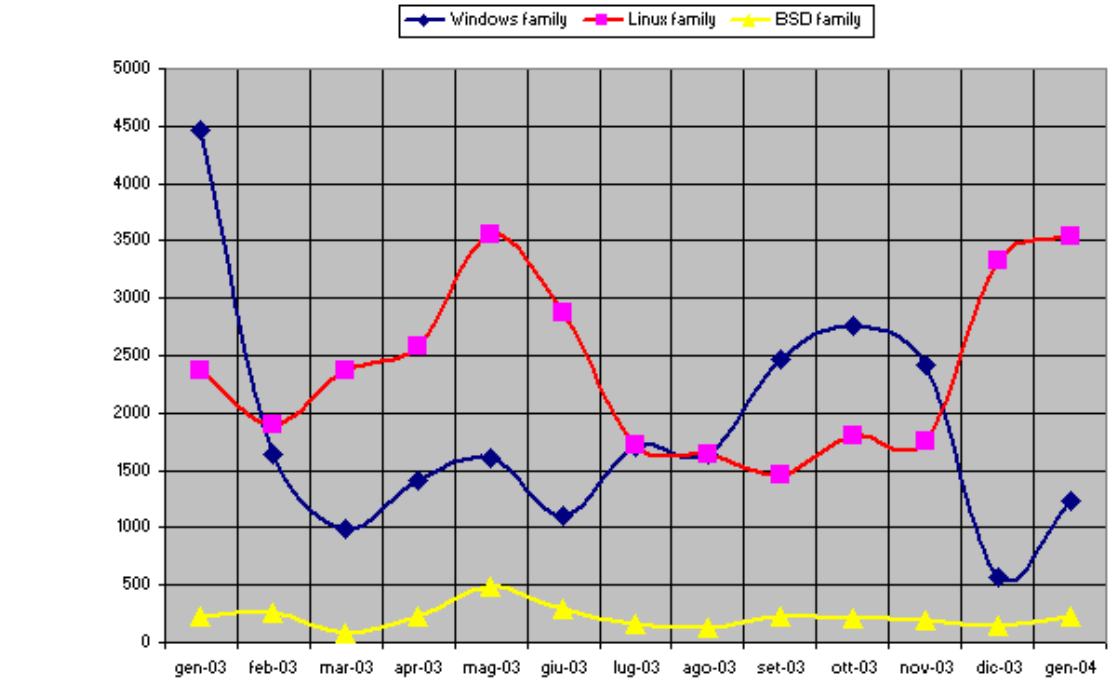
1.4.2 Defacement attacks - Comparison

First we look at the number of defacement attacks on two open source operating systems families, Linux and BSD and one IP-based one, Windows. A defacement attack on a website is one in which the content of a website is replaced with new pages. In most of the defacement attacks the new pages contain religious or political messages, but in reality it can be anything. In the 2004 FBI Computer Crime and Security Survey, among

[12] Eric Raymond, *the cathedral and the bazaar*, O'Reilly, 2001

the organizations covered by the survey, 7% experienced a website defacement attack[13]. Although defacement attack are not the most common form of attacks, it is a form of attack that is very visible outside the organization being attacked, which make them hard not to report.

Figure 1 (below) lists the number successful defacement attacks on website running these operating systems in the period between Jan-03 and Jan-04.



	gen-03	feb-03	mar-03	apr-03	mag-03	giu-03	lug-03	ago-03	set-03	ott-03	nov-03	dic-03	gen-04
Windows family	4458	1634	993	1419	1612	1109	1701	1641	2475	2755	2418	566	1229
Linux family	2371	1900	2373	2586	3551	2870	1716	1636	1458	1797	1753	3333	3533
BSD family	233	255	79	233	492	295	157	130	224	213	190	144	225

Figure 1 - The number of the attacks in the three big families of operating systems: Windows, Linux and BSD (*source: Zone-H.org*[14])

Figure 1 shows that the number of successful defacement attacks on the Linux family was higher than the Windows family in the periods from Feb-03 to Aug-03, and Nov-03 to Jan-04. It is also interesting to note the wide variations in the success rate of defacement attacks on the two open source systems in the comparison, i.e. Linux and BSD families. The Linux family suffered 3-7 times successful attacks when compared to BSD – this might well be a result of BSDs limited market share in comparison to Linux and Windows.

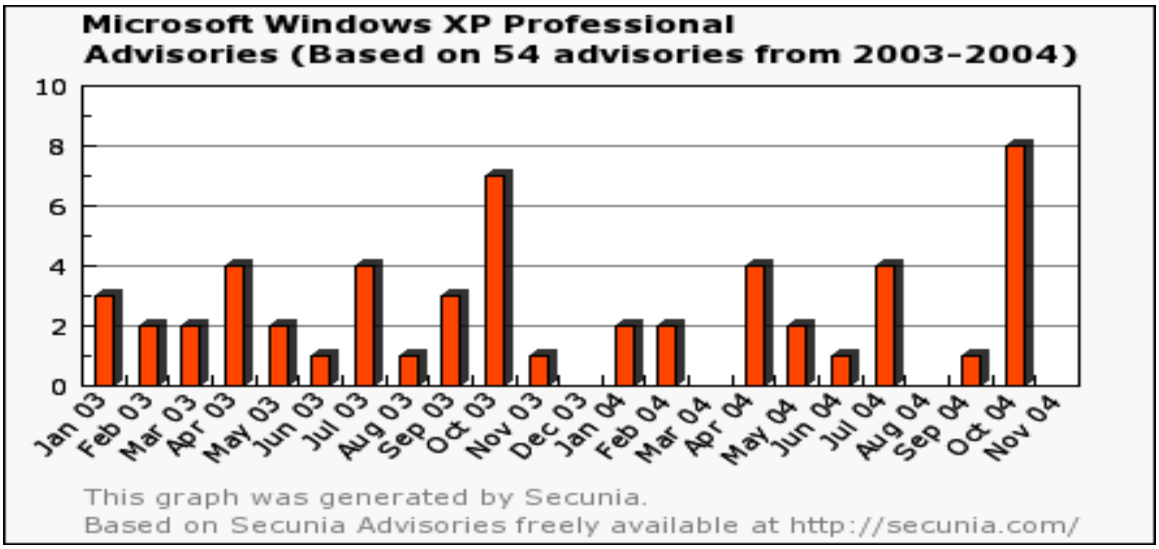
---

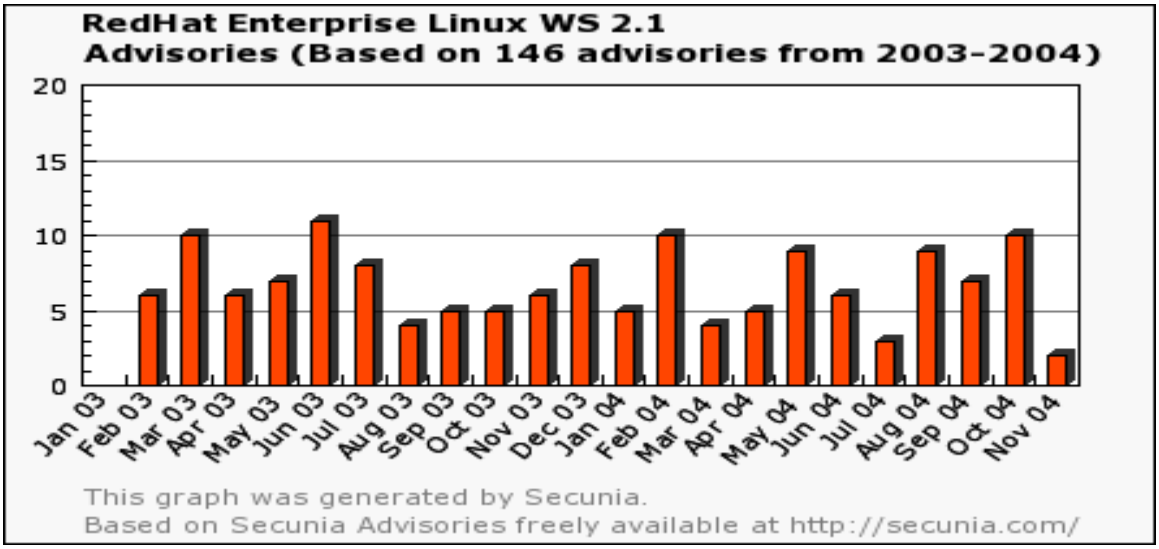
[13] FBI, *the 2004 FBI Computer Crime and Security Survey*, http://www.usdoj.gov/criminal/cybercrime/CSI_FBI.htm

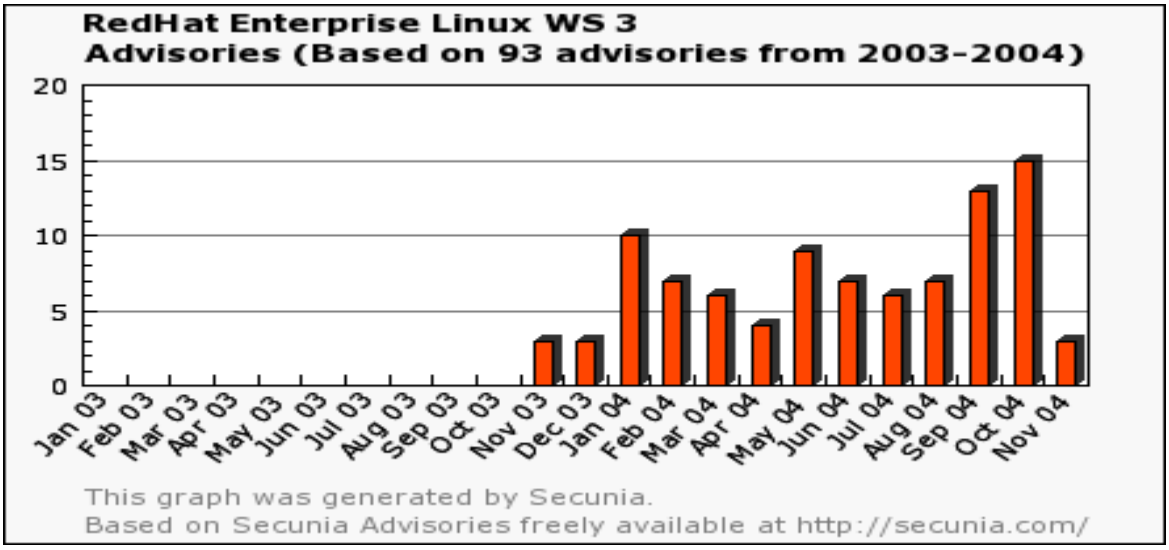
[14] Zone-H, http://www.zone-h.org/en/winvslinux2

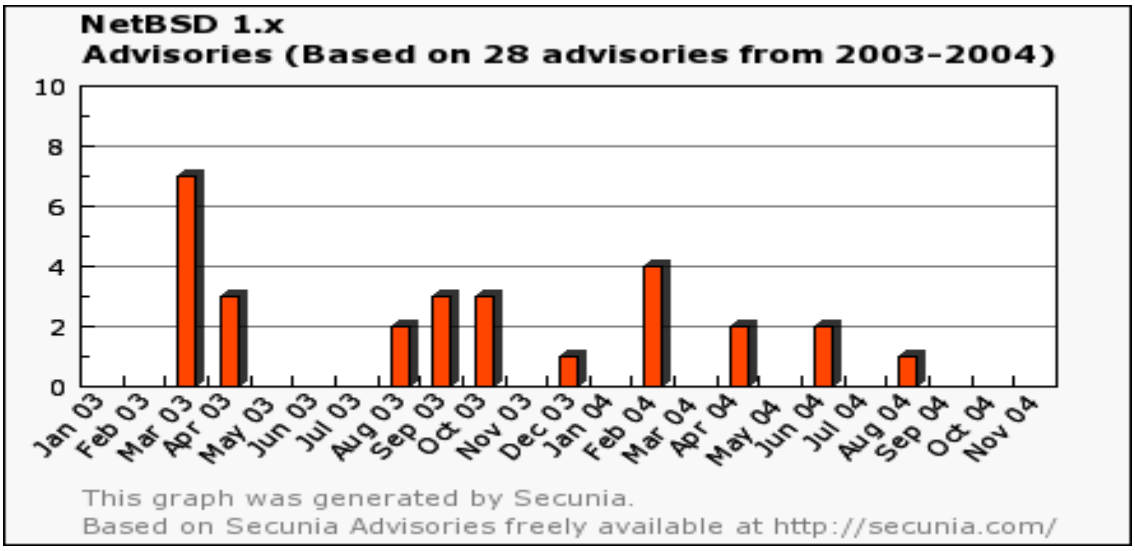
1.4.3 Security Bulletins - Comparison

Another source of interesting information in comparing the security of IP-based software and open source is to look at the number of security bulletins issued by each camp. Before we do so, it is important to mention that as a security metric the 'number of security bulletins' is a highly contested metric by the open source community since it doesn't take into consideration the severity of the report. In our view this is a valid argument if one is evaluating the security of a product to make a procurement decision. Since the goal is to provide a quantitative measure of the number of flaws produced by both development processes just looking at the numbers is sufficient.






RedHat Enterprise Linux WS 3
Advisories (Based on 93 advisories from 2003-2004)
20
15
10
5
0
Jan 03
Feb 03
Mar 03
Apr 03
May 03
Jun 03
Jul 03
Aug 03
Sep 03
Oct 03
Nov 03
Dec 03
Jan 04
Feb 04
Mar 04
Apr 04
May 04
Jun 04
Jul 04
Aug 04
Sep 04
Oct 04
Nov 04
This graph was generated by Secunia.
Based on Secunia Advisories freely available at http://secunia.com/


NetBSD 1.x
Advisories (Based on 28 advisories from 2003-2004)
10
8
6
4
2
0
Jan 03
Feb 03
Mar 03
Apr 03
May 03
Jun 03
Jul 03
Aug 03
Sep 03
Oct 03
Nov 03
Dec 03
Jan 04
Feb 04
Mar 04
Apr 04
May 04
Jun 04
Jul 04
Aug 04
Sep 04
Oct 04
Nov 04
This graph was generated by Secunia.
Based on Secunia Advisories freely available at http://secunia.com/

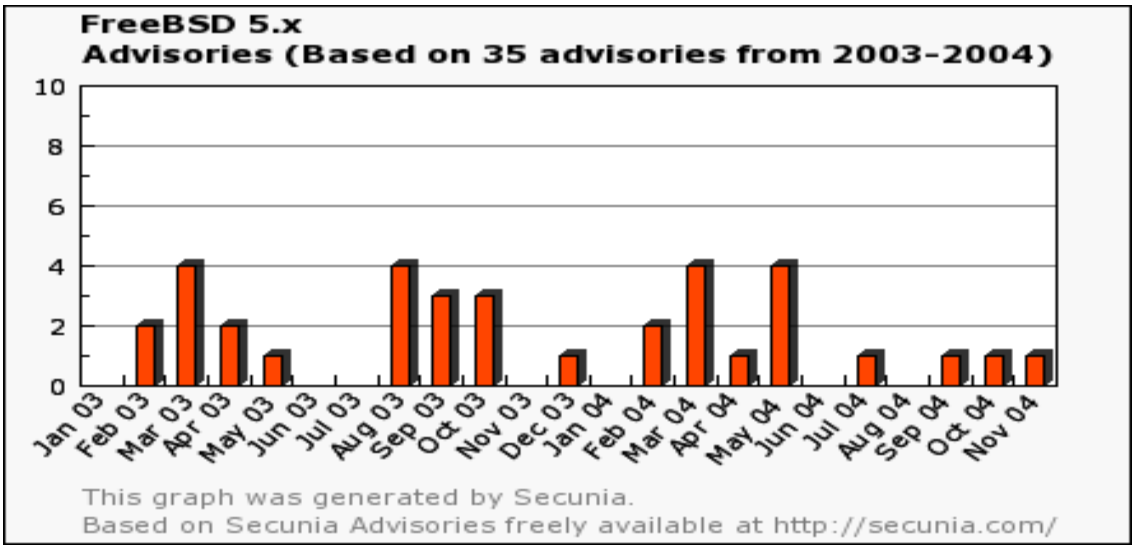

FreeBSD 5.x
Advisories (Based on 35 advisories from 2003-2004)
10
8
6
4
2
0
Jan 03
Feb 03
Mar 03
Apr 03
May 03
Jun 03
Jul 03
Aug 03
Sep 03
Oct 03
Nov 03
Dec 03
Jan 04
Feb 04
Mar 04
Apr 04
May 04
Jun 04
Jul 04
Aug 04
Sep 04
Oct 04
Nov 04
This graph was generated by Secunia.
Based on Secunia Advisories freely available at http://secunia.com/

Figure 2 Number of bulletins issued for Windows XP, RedHat Linux workstation 2.1 & 3.0, and NetBSD 1.0 & FreeBSD 5.0 in the period of Jan 2003 through Nov 2004. (*source: Secunia.com*[15])

As mentioned above the number of security advisories issued is not an ultimate measure of how secure software is. Nonetheless, the fact that the two open source operating system show a wide variation in the number of security bulletin issued - 93 for RedHat Linux WS 2.1 and 35 for FreeBSD with Windows XP in the middle at 54. This data fits the pattern shown in the previous section about defacement attacks.

[15] Secunia, http://secunia.com/product/

1.5 Conclusion

Despite the claims by the open source proponents that open source is more secure, a more close examination of the OSS and IP development processes shows advantages and disadvantages on both sides. The claim of open source intrinsic advantage over "closed source" could not be verified from the examined perspectives.

In our view "openness", being the most controversial aspect discussed, may not not have a big advantage in security. This is evident from the fact that expert "eyeballs" make the difference to the casual developer review. The openness of open source doesn't automatically make it more secure, but it creates an opportunity for motivated individuals to pool together security expertise to do code reviews, security auditing and create tools to help improve security. Two great examples of this are the Sardonix project [16]and static source code analysis project in Stanford University. [17] On the other hand, disclosing source code can be a slight advantage to the expert hacker in reducing the overhead of analyzing issued patches to produce an exploit for un-patched systems. In addition, "openness" could lull developers and users into a false sense of security[18]. Tools might be the answer but it would depend on open source leveraging its potential power to produce them to try to balance the disadvantage they have resulting from commercial having access to open source code analysis tools and more internal ones. Lack of formal testing may constitute a disadvantage to open source but produces an implicit advantage by making developers work in a more responsible manner.

The numbers come in support of findings that both sides exhibit a mixed set of pros and cons. The record of problems found in OSS and IP don´t suggest the superiority of one over the the other when it comes to security. Both open source and IP software have suffered from an abysmal rate of security failures in the last few years. In both worlds the number and sophistication of attacks are on the rise. If software is to meet future needs of business, government and home users, there has to be an order of magnitude improvement in the resilience of software products to attack.

Finally we believe that there is a slew of inherent potential on both sides that could be leveraged. There is also room for hybrid models reaping the advantages of both camps. This might be evident from the hybrid development model used with Mozilla[11]. Companies like Apple and Sun have taken the initiative to freely publish the source code of projects, indicating potentially closer steps toward a hybrid model.

---

[16] The Sardonix Project, https://sardonix.org/

[17] Dawson Engler, *Using Programmer-Written Compiler Extensions to Catch Security Holes*, IEEE security and privacy, Sep 2002

[18] John Viega, *The Myth of Open Source Security*, http://www.developer.com/tech/article.php/626641

2 Economics of Open Source Security

2.1 Introduction

This section investigates the economic impact of security issues and the driving forces within the open source community to combat them. The impact of malicious viruses and worms can cause billions of dollars in damage. To make open source software reliable and usable for consumers, individuals and organized groups have come forward with an interest in improving the security.

2.2 Quantifying the problem – Monetary value of security

A major security exploit often found with desktop software or databases can lead to disclosure of sensitive information. For example, personal details, credit card numbers, social security numbers, etc. With this information, malicious individuals can commit identity theft, one of most widely and increasingly occurring crimes in the United States. Almost ten million Americans had their identity stolen last year alone. According to the Federal Trade Commission[19], the cost to businesses totaled $48 billion in 2002 and out-of-pocket costs to individuals totaled $5 billion.

In addition, over the last decade viruses and worms have infiltrated computers throughout the world via the internet. For instance, the Slammer worm rocketed around the globe disrupting hundreds of thousands of systems, bringing world wide internet traffic to a crawl. Major financial institutions, airlines and other corporations suffered wide spread service interruptions. Accounting for lost working, cost of overtime for IT staff and damage to data systems, Computer Economics estimated the financial impact of the Slammer worm exceeded the $1.25 billion worldwide. In August 2003, Blaster worm and its variants exploited a known Microsoft security flaw, allowing an attacker to control an infected system. The financial loss due to the Blaster worm was estimated to be as high as $2 billion. The most costly attack so far was the LoveBug worm in 2000, exceeding $8 billion world wide.[20]

---

[19] *"Identity Theft Strikes 1 in 8 Adults, FTC Says"*, Technology – Reuters, Andy Sullivan, September 3, 2003

[20] *August 2003 – Worst Virus Season Ever?* (http://www.computereconomics.com/article.cfm?id=867)

2.3 Searching for secure software - Incentives for different parties

2.3.1 Incentives for Consumers

Primary goal for consumers to use a software program is to accomplish a task, e.g. send an email, purchase a gift online, etc. Consumers expect the software to just work, meaning the software program should have security built-in when needed. Although most consumers treat software as yet another commodities, the liability issue is quite different. For example, when problems occur on regular products, manufacturer is responsible for recalling the product and is often liable to any damages caused by the defects. However, there is no such thing in software. Consumers are often forced to follow the contractual disclaimers of liability, which are found on most shrink-wrapped software. Otherwise, operating system vendor such as Microsoft would probably lose billions of dollars each year on liability related lawsuits.

Consumers' awareness of liability towards insecure system has greatly increased over the past few years as number of internet scale virus/worms has risen. As predicted by Nancy R. Mead, software liability will follow the example on automobile, where automobile manufacturers are liable for injuries results from design/manufactured faults in vehicles they produced. [21]

2.3.2 Incentives for Developers

Typically, programmers contribute to open source mainly because of the following incentives:

1) Specific solution that required fixing a bug or customizing an open source program.

In this case, the programmer is in fact driven by the fact that he needs security in the open source program that he is currently using. The open source program can be his mail client, or a server product his company is using. In both scenarios, the programmer is considered a user of the OS program. Because he has knowledge over "non-programmer" users, he has the choice to apply the fix to the program on his own. Obviously, this group of programmers works on a needed basis. They are not going to actively seek out or review the source code for security issues. As number of programmers grows in an open source project, number of free riders also increases. This further lowers the incentives for programmers to fix security problems.

2) Enjoyment and fun of an open source project.

---

[21] Nancy R. Mead, Software Engineering Institute, *Who is liable for Insecure System?*

Although adding new features is fun, reviewing and fixing security bugs is not. And without proper training, it is difficult for leisure programmers to really track down deep security problems. As said by David A Wheeler, "Clearly, it doesn't matter if there are 'many eyeballs' if none of the eyeballs know what to look for." [22]

Providing proper security training is also not as trivial as one may think. For example, during Microsoft's initial Windows security push, all development work has paused for couple months analyzed design, code, test and documentations. [23]

3) Career concern – potential job offers and shares in commercial open source-based companies.

Security is a hot area today and many consulting firms are found to deal with security issues only. Being recognized as a security expert on OS software could lead to potential employment in such consulting firms.

4) Ego gratification and peer recognition.

For many programmers, especially hackers, it is actually more rewarding (in terms of ego gratification) to break a software than fixing it. However, there have been movements by open source groups to recognize people in identifying and fixing security problem in open source software. Although no monetary reward is being given at this point, certain degree of peer recognition is clearly established.

2.3.3 Incentives for Organized Groups

In addition to individual developers, many organized groups have cropped up to provide more credible and accountable security for consumers of open source software. Some groups consist of determined do-gooders, a formation of motivated volunteers with security expertise. Other groups have a business model, paid employees, playing a supportive but profit driven role within the open source community. The driving motivation behind these groups can be vastly different. Consider some groups that have surfaced:

1) Sardonix Project

Consider the largely altruistic approach taken by the Sardonix project. The project is a security portal, seeded through DARPA money. It is a centralized location for open source community members to archive security audits of prominent open source software. Anyone can register and contribute security audits. Contributors are not financially rewarded; instead, they are given points weighted on the coverage of their

---

[22] David A Wheeler, *Secure programming for Linux and Unix How To*

[23] M. Howard And S. Lipner, *Inside the Windows Security Push*, IEEE Security & Privacy, 2003

security audit. If a member audits software that is later found to contain security issues; he or she is docked points. In this way the point system is a display of status; more points means greater peer recognition, a common driver for many contributors.

2) Linux Security Audit Project

Most often the contributors of security review groups are united around a particular open source project. For instance, the Linux Security Audit Project is aimed specifically at identifying issues in the Linux operating system. Membership is open to all and individuals contribute through an online distribution list which is monitored by Linux component owners. People are encouraged to discuss security issues and recommend specific areas for others to review. The incentive for the project is to provide a community center to house the collective efforts of its members. Prominent figures within the community are responsible for assessing the risk of security flaws, rolling out fixes, and posting security advisories.

3) The Apache Group

The Apache Group, responsible for the Apache web server, works in a similar way; however they favor a private reporting process. In order to prevent hackers from exploiting vulnerabilities before fixes are available, participants send email to a private distribution list. Once received, a core group of volunteers with security expertise analyze the issue and produce a fix. The reasoning is that keeping information in a private forum means less opportunity for exploitation.

4) OpenBSD Project

Most open source efforts are reactive in their attitude toward security. However, security experts agree that a proactive attitude is the best approach for dealing with security. Some open source projects, such as OpenBSD have successfully adopted this attitude into its philosophy. The project's proactive emphasis on security has rewarded it with a reputation for being one of the most secure operating systems available. OpenBSD has a team of about a dozen volunteers with significant security backgrounds, reviewing each lines of code in the operating system components. They review for bugs instead of specific security flaws because security vulnerabilities are often enabled through otherwise insignificant bugs. Security holes found in competing BSD variants were found and resolved in OpenBSD, often months earlier, through its code reviewing process. OpenBSD, created with the intent of being the most secure operating system available, has largely succeeded through its proactive security attitude.

5) Consulting Companies

In addition to altruistically motivated groups, monetary incentives have created a market of consulting companies that play a supportive role in the open source community. Many consumers of open source software are willing to pay for a secure solution. Some consultants offer a complete set of services ranging from hardware and software installation and ongoing support, to security testing and policy configuration. The capabilities and skills offered by such commercial companies must be assessed on a case

by case basis. There is no standard to rate and compare how well these companies provide a secure solution so the quality of service is hard to gage. Security is a popular buzz word for consultants; however, some of these companies are so small it seems questionable they would have resources and expertise to offer credible solutions. In contrast to the altruistic groups above, these firms are primarily driven by profit.

There is clearly value gained by collectively organizing developers with an interest in improving security. Different incentives such as point reward systems, public and private recognition, and reputation amongst peers have been used by groups to improve the security of open source software. Profit motivated companies have also surfaced to provide tools, and complete hands on support for consumers of open source software.

3 Open Source and Government

3.1 Government Policy on Open Source

3.1.1 Introduction

It is clear that security issues are the major concern in every country where software is controlling critical data and processes.

Governments have started to see the benefits of the new open source movement. It leverages contributors around the world to work on projects, greatly reducing the development cost. There are two other major benefits. The first is more eyeballs on code, meaning anyone can view the source code and contribute to identifying and fixing bugs. The second is that because it is open source, it is much less likely ¨backdoor¨code will exist since all code is freely seen. With proprietary code, only the hard to understand binary executable is available for inspection.

Governments of different countries are actively making policies about the new OSS movement. Except for the United States (US), countries are economically and politically motivated to promote OSS. The US is taking a neutral stand, while still taking necessary and effective actions toward establishing a security standard for all the software, where open source or not.

In this section we will discuss different government policies relating to open source, and specifically the implications relating to security.

3.1.2 United States

The US government does not promote open source or proprietary software. It does not believe one is more secure than the other. Their current policies take into consideration of its role as policy maker, user and contributor of software security.

Government Regulation?

Although there was an unsuccessful initiative by some congressmen to reject OSS on the basis that lack of profit incentive would kill innovation for software security, it was unsuccessful. The US government is not interested in regulating the software security industry; it would rather see the power of free market at work. It has taken a leadership role by being an example of security internet usage.

Certification Program-NIAP

The US government initiated the National Information Assurance Partnership (NIAP) effort to meet the security testing needs of both information technology (IT) consumers and producers. The long-term goal of NIAP is to help increase the level of

trust consumers have in their information systems and networks through the use of cost-effective security testing, evaluation, and validation programs. Both Open Source Software and proprietary software must pass the tests to be certified, at they own expense. The large cost of going through the certification process is not a problem for proprietary software vendors but a challenge for open source projects.

IT Security Planning

The US government enforces and encourages IT security among its agencies by mandating IT spending level on security. All its agencies are encouraged to have their own plan for IT security, this does not necessarily mean using open software or proprietary software, only demonstration of a focus security and cost effectiveness. All the software in government must be NIAP-certified.

Contribution to OSS security

The US government not only sets an example but also contributes useful tools and source code to the general public. NIAP provides testing programs for any non-government deployment. National Security Agency (NSA), the US governments´ department responsible for IT security, developed a Linux-based application, which enhances its security by role-based access restriction and type enforcement. This guards against buffer overruns. This application is based on Linux for three reasons:

- Although taking a neutral stand between OSS and proprietary software, the Bush administration has decided US government should not miss the benefits of OSS movement.

- Linux is a mainstream platform now.

- Open source is viewed as a better way of attracting more talented reviewers to contribute and review potential security holes.

The end product is released to the general public under GPL license and the Linux community is encouraged to contribute.

Conclusion

The US government has decided to push a focus on security and innovation in both the open source and proprietary software worlds. It is actively involved as user and contributor of OSS.[24]

---

[24]Q&A: Does the U.S. government have an open-source security plan? *Linux World*, December 11, 2002 http://www.linuxworld.com/story/32813.htm

3.1.3 Germany

Besides the US government, there are many other governments around the world that have actively embraced the open source movement. Some proposals include government subsidies for research and development (R&D), standardization on, and procurement preferences of open source software. The European Parliament, for instance, adopted a resolution in September 2001 that calls on the commission and member states "to promote software projects whose source text is made public". The German Bundestag is considering legislation that would require government agencies to use open source. Former French Prime Minister Jospin created an agency whose mission will be to "encourage administrations to use open source software and open standards".

Specifically noticeable are countries in Europe like Germany, France, Italy, Spain and also countries in other continent like Brazil, Venezuela etc.

Germany, for instance, is the most active national government in this field. It has undertaken concrete initiatives to promote open source. On March 14, 2002 for instance, the Council of Elders, a joint deliberative body whose task is to manage the internal affairs of the Bundestag, arrived at a decision on the new Bundestag information technology ("IT") environment. It decided to follow the LuK (an eleven member committee comprised from major political parties) recommendation to install Linux on approximately 150 servers. A national open source competency center will also be set up to serve as a nodal point. It will provide necessary technical infrastructure, discussion forums and a marketplace for the open source community and its users.

Open Source make software more secure?[25]

Marking the software open source alone does not mean that it is secure. The rule is, users, as well as programmers, will not able to ascertain whether a specific program is secure. Only a small group of specialized experts will possibly able to do so, and that only after a detailed study. In spite of the questions which remain open, the fundamental precondition for evaluating the security of software is certainly for its source code to be made open.

This philosophy has been echoed by comments of Siegmar Mosdorf -- the German Secretary of State in the Federal Ministry for Economy and Technology - .

Open Source Policies, Center for Strategic and International Studies, September 2004
http://www.csis.org/tech/OpenSource/0408_ospolicies.pdf

[25] profiling of open source developers; huge number is from Europe
http://www.techweb.com/wire/story/TWB20001101S0016
Mosdorf http://www.internetnews.com/bus-news/article.php/6_408271
eEurope http://www.egovos.org/Resources

"Security through Obscurity" is the motto of yesterday. The slogan must today be "Security through Transparency". The German federal government expects an increase in security in data processing and data communication because of the open source movement. This is because if the source code of a program is clear and visible and can be checked by experts, then security is considerably increased.

According to Mosdorf, the German federal government expects an increase in security in data processing and data communication because of the open source movement. This is because if the source code of a program is clear and visible and can be checked by experts, then security is considerably increased. "Security through Obscurity" is the motto of yesterday. The slogan must today be "Security through Transparency". For this reason the Federal Ministry for Economy and Technology is supporting the development of an encryption product on an open source basis. This initiative is not about putting certain businesses under pressure. The competition in security matters will benefit the IT sector, says Mosdorf.

There are many counter points as well. Open Source; because source is available, it is easier, for attackers to apply theoretical research to trying hacking the system. Community contribution, require a higher discipline and higher standard on coding and security education…

So, so far, whether open source make software more secure is debatable, but, it is clear in Government's mind, it definitely help moving into a right direction with the source availability and community involvement, and this is especially strong in Europe and populated into other continents as well.

Economic Motivation?

Over the last twenty years, most governments have chosen to increase their reliance on market forces to govern the production and distribution of goods and services. Some previously communist countries, such as Poland, have sharply reduced centralized planning, privatized major national industries, and introduced market competition. Some capitalist countries, such as the United Kingdom, have reduced their reliance on government regulation and attempted to increase the scope of market competition. Policymakers are generally more skeptical than they were twenty years ago about the wisdom of having governments control markets. So, it is bit surprising to see so many countries entertaining policies to promote a particular, namely open-source, method for producing and distributing technology. This is surprising because governments have recently been making fewer efforts to endorse particular methods.

However, to some degree, this is not surprising in other countries except US, because Open Source software presents the most realistic, and sometimes the only real technical and economical alternative to Microsoft products.

"I am convinced that open source development can form the European base model in the information age". With these words, the German Secretary of State in the Federal

Ministry for Economy and Technology, Siegmar Mosdorf, definitively voiced the backing of the German Government at the Linux Day 2000 of the open source movement.

The German government's national interest has yet another argument in favor of open source. As Professor Lutterbeck stated in his report commissioned by the federal Ministry of Economics and Technology, "new and not yet discussed in Germany are the figures of the worldwide occurrence of open-source developers. They show that German developers firm the second largest group. On the whole, European developers are the predominant". He concluded that the open-source area not yet dominated by the United Stats is of great economic importance for Germany. In its resolution on "Germany's Economy in the Information Society" on November 9, 2001, the German Bundestag also stated that open-source software should be considered a special opportunity for the European software industry and should not be missed. Helping domestic programmers was also explicitly stated as "one of the big reasons" for the GPL mandate in Venezuela.

The end results of the dominance of Open Source development in Europe has clearly shown in the studies of Paul, Jones, director of ibiblio.org.

Paul Jones, director of ibiblio.org. Jones, a University of North Carolina journalism and library science professors and director of the UNC MetaLab and ibiblio, studied the demographics of the Linux developer community, counting the people who were contributing to open source. It turns out that more than 50 percent were people adding just one or two pieces to the growing project. 250,000 developers worldwide are involved in open-source work, contributing code, writing documentation, or tracking bugs. Jones' research showed German contributions to open-source software were greater than people with .edu e-mail addresses and also greater than the number of people with.org and .gov e-mail addresses. Combined the entire EU, it is greater than all the dot-coms. "The demographics of contributors reveals a strikingly strong European influence within the Linux community", the researchers wrote.

Conclusion

Clearly the national interest is on top of the force behind the Open Source adoption and development push in Europe and other counties. It is a clear a golden opportunity for those countries to try to overturn the US software dominance. And this is clearly to us, a major motivation than pure security concern with existing proprietary software.

3.1.4 China

National Security

Different governments have different reasons to believe Open Source or proprietary software is more secure. Governments outside of US, China as an example, are concerned on the dominance of Microsoft and even US in the proprietary software industry and their dependency on US-made software for their critical operations for national security.

They are concerned about backdoors hidden by American manufacturers. An article on China Economic Times expressed the concern of China government particularly because the users don't know what the proprietary software is actually doing or whether the data is being inappropriately shared. Microsoft addressed their concerns by going through required tests, revealing part of the source codes and allowing China government to build the binary codes in their environment from the source codes reviewed. Despite all the efforts by proprietary software vendors, the balance is still hard to achieve between national security China government and IP concerns of the vendors, which gives OSS an upper hand. November 2003, China government proposed to contract a US company for deployment of OSS on 1 million computers.

Another concern is potential software export restriction from US and its allies in the form of Coordinating Committee of Multilateral Security (COCOM). Despite of the disbandment of COCOM in 1994, US still has laws restricting high-technology export to certain countries, including China, which encourages the rapid growth of Open Source software development in those countries.

Economic and Political Motivation

Belated legislation and lack of enforcement on IP protection makes foreign-made proprietary software virtually free, which results in the dominance of foreign-made proprietary software, almost non-existence of domestic software industry and deterrence of the progress of Open Source movement. This is changing with the tightened copy right legislation and law enforcement required by the World Trade Organization agreement, which motivates China towards developing its own platform. China developed its version of Linux called Red Flag Linux in 2003 and plans to continue development with Korea and Japan.[26] Ministry of Information Industry R & D Approved MII established the Open Source Software Promotion Alliance to encourage the development of China's OSS industry.

China's government doesn't regulate whether the government or the general public should use OSS or proprietary software. It does set policy on software purchase by government agencies. There is a strong nationalism sentiment among Chinese general public right now for their pride as a rising economic power in the world and dissatisfaction of being far behind in the high-technology game. China's government is walking a fine line between being a fair player in the global market and protecting its domestic IT industry. Currently, China government doesn't specifically require OSS but promotes purchase of domestic products, which are more likely OSS. China State Council Preferences Approved August 2003 mandate that all ministries buy only locally-produced software at the next upgrade cycle. [27] Microsoft's efforts to get around the

---

[26] Korea, Japan and China Plan to Develop Secure Platform Together
http://tech.sina.com.cn/it/e/2003-08-31/1557227518.shtml

[27] Cooperation of Government and Industry
http://www.ccw.com.cn/news2/soft/htm2004/20041101_0978Q.asp

barrier by establishing China division were countered by China's think tank's proposal of requiring more than 50% value addition in China.[28] Municipal governments followed suit.[29] In 2001 Beijing Municipal government awarded six contracts to Chinese vendors, including a deal of 2000 desktop OS license to Red Flag Linux.

Conclusion

As an effort to balance its national security, economical and political needs and open-market reform efforts, China government promotes OSS by direct funding of OSS development and preference of OSS software for government purchase decisions.

3.2 Impact of Roles Government play on Open Source

3.2.1 Introduction

There are many hats Government carry: they are, at first, the policy maker; they are also both the software consumer and producers. Because of these roles, their policy can make huge impact on the Open Source movement, and as well the overall security of the software industry.

3.2.2 Role as Policy Maker[30]

One of the roles Government plays is the policy maker. It provides the regulations and policies for the welfare of the people and the nation. It has been the trend of last 20 years that most governments have chosen to increase their reliance on market forces to govern the production and distribution of goods and services.

There is a consensus among economics about the principles that one should follow in determining whether an intervention is desirable. First, need to identification of significant market failure – a significant flow or breakdown in market process, which prevents competition from giving consumers the greatest possible benefits given scarce

---

Software Purchase Policy
http://www.grp.com.cn/zyzx/view.asp?id=1636&cc=1&counter=1
Protection of Local-made Software http://tech.sina.com.cn/it/2004-08-26/1441412926.shtml

[28] Microsoft China Not Recognized as Local Software Vendor
http://news.xinhuanet.com/it/2004-09/01/content_1941212.htm

[29] Promotion of Linux by Municipal Government
http://shenzhen.ccw.com.cn/news/200409/0920_01.asp

[30] Secure Linux by NSA http://www.nsa.gov/selinux/
NIAP Program http://niap.nist.gov/

resources. Second, government intervention will actually improve things (i.e. make consumers better off).

Economists studied the software market, and concluded that there is no market failure.

They measure using two major methods: conventional measure of industry concentration – by total share of sales accounted for by the four largest forms. In 2000, the four largest firms in proprietary software industry accounted fro 26.7 percent of total revenue, while over 47 percent of all manufacturing have a four firm concentration ratio greater than that of software industries. The second method is using Herfindahl-Hirschman Index (HHI) for measure of industry concentration. HHIs can range from 0 to 10,000 (a monopoly with 100percent of the market share). In 2000, the HHI for the software industry was 244, a relative low HHI when compared with other familiar industries such as automobiles (2,506) or breakfast cereals (2,446).

Software is a very competitive market, results a great deal of turnover. Study shows that five of the top ten companies in 1990 did not make the list in 2000.

Based on these facts, it shows clearly to the Government that they don't need to intervene the market since there is no market failure.

However, there are several concerns Governments have as the policy makers.

1)The digital era presents significant opportunities and real risks for developing countries. One risk is being sidelined from software trends that drive the increasingly digital global economy. As developed economies increasingly crease networked purchasing and production system that depend on advanced ICT infrastructure, countries that are not connected on favorable terms, and business within those countries, may be deeply disadvantaged. The decisions Government make about procurement, standard setting and ICT adoption, technology investments, and training are critical. Open Source, not only can reduce the burden of limited IT cost, but, it can also allow more domestic talent participating in the development of local software. Clearly there is the motivation to upgrade the country's human resource capacity and technological skill base.

This not only related to the developing countries, it also related to those developed countries in Europeans etc. Since U.S are the dominant in software business. Worldwide commercial software is a $171 billion business based on revenue, and U.S-based produce of $90 billion in 2000. Top 10 Software companies ranked by revenue and market capitalization are all U.S companies as shown in table 2:

Ranking	**Company**	**Annual revenue** *(millions of $)*	**Market capitalization** *(millions of $)*
1	Microsoft	31,375	260,000
2	Oracle	9,487	63,400
3	SAP	7,700	32,300
4	Computer Associates	3,083	12,400
5	VERITAS	1,531	10,100
6	Electronic Arts	2,489	9,300

7	Intuit	1,373	9,000
8	Adobe Systems	1,194	8,000
9	Symantec	1,328	6,600
10	PeopleSoft	1,949	4,700
11	Competition	38,445	28,582
Total	All	69,954	444,400

Table 2. Top 10 software companies, ranked by revenue and market capitalization[31]

So, there is a real motivation for Government policy marker to figure out a way for boosting their own domestic software industry and break the current trend as technologically dependent on a few major software suppliers located in other countries.

2) Open Source can also have an anti-monopolistic effect on the IT market and industry.

3) Security of public data is a leading concern of Governments, particularly in the wake of recent world-wide computer virus attacks and growing fear of cyber terrorism and cyber crime, as well as spy ware. At a minimum, introducing diversity into the base of functioning software code reduces the possibility of catastrophic failures caused b viruses that attack a software monoculture. Finally, because Governments can't choose their customers or citizens, it follows that they should not oblige them to use costly proprietary software and closed data formats. The need for pen public data formats is directly relevant to calls for increased accountability and transparency in public sector governance. The use of standard and open formats guarantees free access. If one is to guarantee the permanence of public data, the usability and maintenance of software should not depend on the goodwill of suppliers or on conditions imposed by them in a monopoly market. At a fundamental level, nations must, in order to guarantee nations security, be able to rely on systems without elements controlled at a distance.

Other developing countries have also expressed dissatisfaction with the proprietary software development and marketing model, in particular pointing to the negligible influence them, as "smaller" customers, have on how software develops. FOSS is expected to provide more flexibility and allow more autonomous input into software development. This can be conceived of as an ownership issue: developing nations desire an opportunity to articulate their software needs and participate in the innovation process as end users of software products.

4) With increased emphasis on and pursuit of Intellectual Property Rights enforcement at the international level, the choices available to software users are becoming more distinct. All efforts by international proprietary software producers to decrease piracy in fact improve the fundamental conditions for increased adoption of open-source software.

---

[31] UNCTAD estimates based on data from Yahoo Finance http://finance.yahoo.com and Financial Times Market Data and Tools http://www.ft.com

All above triggered Government Policy towards Open Source Software. The results are amazing. In the last few years GNU/Linux has increasingly penetrated both the high and low ends of the enterprise market for operating systems. Nearly 40 percent of large American companies and 65 percent of Japanese corporations use GNU/Linux in some form, and it may now run as much as 15 percent of the large server market overall. A study from October 2002 found that 59 percent of software developers surveyed internationally expected to write applications for GNU/Linux at some point during the next year. The EU-sponsored FLOSS survey found 43.7 percent of German companies and 31.5 percent of British companies using FOSS. It is notable that, according to several suits, Internet service providers (ISPs), large companies, small companies, and CIO's in financial services, retail and the public sector all believe that GNU/Linux use is set to increase rapidly both in their own organizations and in the market as a whole over the next several years.[32]

In summary**,** even though the study clearly shows there is no market failure in software industry. Countries outside the US are making all the policy to support Open Source movement. They use this as chance to encourage the domestic development, and also treat the Open Source as the way to compete, and gaining on market share which currently dominant by US software industries. This in author's view will have a huge impact to the market and industry and make Open Source is even a more formidable force. Although whether this will increase the security in software or worse, since we don't have enough time or data for the analysis, clearly, it will flourish with Government as the adopter and strong supporter.

So far, the U.S. government has the right policy; it doesn't promote Open Source nor promote Proprietary Software. It lets the market to choose the winner. At the same time, as a policy maker, it continue the role of providing framework, process, resources, tools, and regulation to urge the industry move into more secure computing environment. U.S. should not ignore the Open Source movement, and it should provide the enough support to flourish the effort, so that it guarantees the long lasting the dominance of U.S. software industry and push the industry towards more secure computation by facilitating more reaches, process and matrix.

---

[32] E-commerce and Development Report, UN Conference on Trade and Development, 2003 http://www.unctad.org/en/docs/ecdr2003ch4_en.pdf

3.2.3 Role as User[33]

Governments are major software consumers. Their decision are not only based on the features and quality of the products (such as security) and cost effectiveness, but also on national security and the economic impact to domestic IT industry.

Security

As discussed in the examples, governments have their own reason to believe whether OSS or proprietary software is more secure. Representative arguments for OSS security are no hidden backdoor in OSS threatening national security and more eyeballs checking for security holes, discussed in section 1. Polymorphism of Open Source software also makes specialized software simpler than the model of one-fit-all code base plus configuration. Some governments such as Denmark believes lower complexity reduce security risk, which is true. However, forking also means fewer eyeballs checking the specialized codes. Rapid identification of security holes doesn't translate to adequate handling at individual user level, lacking the commercial channels of proprietary software vendors. Adequate consulting and maintenance service are required.

Some governments such as UK government and Peru government intentionally introduce platform diversity to isolate or reduce the possibility of catastrophic failures caused by virus/worm attacking a software monoculture. This is a security practice at deployment level.

All these present challenges to proprietary software vendors:

- they are hard to be modified or extended for specific needs of governments
- they may not be compatible with OSS applications while public service is required to provide access from any platform for fairness and diversity is preferred

---

[33] E-commerce and Development Report, UN Conference on Trade and Development, 2003 http://www.unctad.org/en/docs/ecdr2003ch4_en.pdf

Open Source Policies, Center for Strategic and International Studies, September 2004 http://www.csis.org/tech/OpenSource/0408_ospolicies.pdf

Open Source in e-government, Danish Borad of Technology, October 2002 http://www.tekno.dk/pdf/projekter/p03_opensource_paper_english.pdf

Open Source Software Use in UK Government http://www.govtalk.gov.uk/documents/OSS%20Policy%20draft%20for%20public%20consultation.html

It has to be pointed out proprietary software vendors are opening their source codes to certain users, including foreign governments. Microsoft is promoting a "share source" program to address all the above concerns. It opens up some of the source codes for security review and code extension for special needs and compatibility with OSS applications. IP right concerns limit the scope of opened source codes and put them in a disadvantaged position.

Cost Effectiveness and National Economy

It is still debatable which type of software has lower total cost of ownership. Microsoft conducted a study showing proprietary software actually costs less than their Open Source counterpart with lower cost of configuration, extension, management and maintenance. However, the initial cost of proprietary software is based on US standard and much higher measured to the income and labor costs in developing countries. The lower labor costs in developing countries and the strong Open Source community and talent pool in European countries make the follow-up development, customerization and maintenance cost lower than in the US.

Another consideration is that for countries with nascent IT industry, it could be more beneficial to spend the software purchase money on developing its own software, most likely based on OSS, to train and grow their own talent pool with spillover effects on their IT industry.

Impact

Government adoption of Open Source or proprietary software has huge impact. First of all, governments are big customers or even the single biggest customer in many countries. Secondly, the compatibility issue between OSS and proprietary software make the governments' decision carry strong network effect. The eEurope 2005 Action Plan by EU promotes and recommends OSS to states within government body and public administration. The compatibility issue may not be that significant right now, given most of the migrations are on the server side because of the maturity and dominance of Microsoft on the desktop software. But government intervention may break the balance, though. European Union, especially France and Germany governments, have a plan to migrate 5%-10% desktop to OSS platform. The EU funded a research project (the IDA Open Source Migration project) to provide guidelines to migrate to OSS-based office applications such as document processing, email, calendaring and other standard application. Compatibility issues will pressure private companies, non-profit organizations and individuals who interact with government and/or public administration to OSS. With their eyes on security, governments provide big incentives for OSS security.

3.2.4 Role as Contributor[34]

Governments have special security needs for government operation and public service, and the responsibility to enhance overall network security for the general public. So they often also play as a contributor. What's more, one of the most significant characteristics of OSS movement is that contributors are mostly users since OSS movement partially originated as "scratch-the-itch" efforts. Inevitably governments become OSS contributors from users.

Developing a Secure Platform

Some US government agencies such as Department of Defense have their own development of security software. However, they are mostly classified so the general public can not benefit. National Security Agency developed a Linux-based application, which can be integrated into any Linux platform to enhance security, and released the products to general public under the GPL license, an effort to contribute to network security for the general public and to attract wide contribution and code reviews from OSS community.

As discussed in section 3.1.4, a lot of countries are behind in the software industry. Existing successful OSS products and community provide an easy entry for them to catch up. First of all, the open source codes are free to use so it reduces the cost of initial development. Secondly, the codes are open for highly specific customerization. Thirdly, the Open Source communities provide worldwide technical support.

The direct involvement of government also provides incentive, education and training for their talent pool, which may incubate their IT industry and enhance network security for general public.

Funding Security Research

Government funding for academic internet security research is growing in all the countries, providing incentive to both proprietary and Open Source forms. Due to strong influence and presence of Open Source movement in the academy and/or the strong preference of OSS by the governments, the efforts are leaning towards Open Source. For example, the European Union passed a directorate in 2003 to urge funding for Open Source encryption software.

---

[34] E-commerce and Development Report, UN Conference on Trade and Development, 2003 http://www.unctad.org/en/docs/ecdr2003ch4_en.pdf
Open Source Policies, Center for Strategic and International Studies, September 2004 http://www.csis.org/tech/OpenSource/0408_ospolicies.pdf

Certification Programs and Testing tools

Some governments, such as US government (NIAP program) and China government, develop security measurement programs and make the tools available to general public, providing measurement and incentives for overall software security.

3.3 Conclusion

Government policy and involvement have created and will continue changing the overall picture in software industry and related the security implication.

- It is changing the development process of OSS. As discussed in section 1, OSS development process is more of a grass-root effort on voluntary ground, although some corporations may form a core management for the process. Governments' direct involvement changed the picture. Governments fund OSS projects, recruit talents from academy, private sectors for development and OSS community for testing and bug reporting, manage development and bug fixing process and provide testing programs. This hybrid model injects the benefits of proprietary software development process to OSS process.

- As discussed in the examples of Germany and China governments, governments' preference provides economic and political incentives for OSS and the related security.

- It will change the landscape of how proprietary handling security, and availability of source and interaction between OSS software.

- Although only domestic policies were discussed, they all have ramification effects in the global markets. If released under Open Source licensing, any contribution by government will be available to the world. More significantly, government policy of an individual country can R&D, distribution and sales of IT companies, who increasingly rely on the global market for talent pool, labor and sales.

4 Conclusion

The security of open source is an important aspect of OSS that is not fully explored yet. From the perspective of the development process, there are pros and cons in both open source and proprietary software. The "openness" of open source, which is deemed by some as an absolute security advantage, was closely examined and argued not to be so in all cases. The data comes in support of this argument showing no clear advantage to one camp over the other. When it comes to the development process we believe the advantage would be for the side that leverages its strengths.

Software security problems have caused the world billions of dollars. Different parties in the open source community have also shown clear interests to improve security in OSS. The costs and incentives together provide huge economic support for security in open source.

Governments around the world are strongly behind the OSS movement, which will have a huge global impact in the near future. OSS is here to stay as a fierce competitor to proprietary software. Therefore, OSS movement will bring the overall security to a higher level in software industry.

Contributions

Section Number	Title	Contributor
1	Aspects of the Development Process and Impact on Security	Mohammed & Ahmed
1.1	Introduction	Mohammed & Ahmed
1.2	Open source Development Process	Mohammed
1.3	Development Characteristics and Security	Ahmed
1.4	Security Statistics	Mohammed
1.5	Conclusion	Mohammed & Ahmed
2	Economics of Open Source Security	Winfred & Eric
2.1	Introduction	Winfred
2.2	Quantifying the problem – Monetary value of security	Winfred
2.3.1	Incentives for Consumers	Winfred
2.3.2	Incentives for Developers	Winfred
2.3.3	Incentives for Organized Groups	Eric
3	Open Source and Government	Lin & Man
3.1.1	Introduction	Lin & Man
3.1.2	The United States	Man
3.1.3	Germany	Lin
3.1.4	China	Man
3.2.1	Impact of Roles Governments Play on Open Source	Lin & Man
3.2.2	Role as Policy Maker	Lin
3.2.3	Role as User	Man
3.2.4	Role as Contributor	Man
3.3	Conclusion	Lin & Man
4	Conclusion	All