Open Source Security

Imran Ali
Derek Cheng
Asad Jawahar
Osama Mazahir
Jared Smelser

Table of Contents

What is Open Source?

Introduction

Just as the internet and World Wide Web find their origins of creation in attempts to encourage the dissemination of information between users, the Open Source community finds its own origin in a similar wellspring. Open Source Software and the majority of its variants is an attempt to be just that, an open arena of software development stemming from a non-proprietary operating system. This openness allows multiple programmers and users to build off one another's ideas and creations, using the source code as a common structure from which to start. Over the years, a set of criteria for the distribution of this type of software has set the boundaries from which this movement has solidified and prospered.

The following is a cursory examination of the Open Source Community through its inception to its present day operations, primarily centering on GNU/Linux, its most successful operating system. This will include a brief history of the Open Source movement, to better understand the overriding philosophy that both guides and fuels this community. Further we will explore how Open Source software is developed and distributed and in doing so shed more light on how Open Source has become so significant within a short period of time.

A Brief History

As mentioned above, the internet and WWW each began as open venues in which mostly academics and some private institutions shared information between one another. This same collegial atmosphere prevailed within the young computer science communities of the sixties and seventies. While computer hardware was mostly proprietary (as most business models of the time saw this as the only real revenue stream from this market) software was passed freely from programmer to programmer, between private industry and universities to lone enthusiasts. Within this community, AT&T's Bell laboratories developed the UNIX operating system. UNIX was created to be a transferable, multi-tasking and multi-user operating system. It would be from this operating system that the Open Source movement would begin to take shape, although the Open Source moniker did not arrive until the late nineties.

Open Source began as the Free Software Movement, when in 1983 its founder Richard Stallman, an MIT researcher and early programming pioneer published the GNU project. GNU is a UNIX-like operating system, not under the licensing copyright of AT&T's UNIX, setup to be a building block from which programmers would have free access to build on. Stallman created GNU in response to the restrictions the computer industries were beginning to place on proprietary code. He intended that users be free to study this source code, modify it when wanted and produce it as long as the subsequent software was again free to other users. Stallman wanted a return to the collegial programming community of the sixties and seventies where an open marketplace of ideas

would foster growth and innovation. To help this take place he created the Free Software foundation and the General Public License (GPL); this non legal contract would give licensees the right to copy, modify and sell these programs as long as they granted downstream rights to future programmers, Stallman called this “Copylefting”.

While during the eighties and nineties open source code systems gained prosperous inroads with UNIX programs such as Apache, Perl and Sendmail, (see chart below) it would be with the creation of GNU and the subsequent Linux Kernel when open source programs became market forces. The kernel is a program on UNIX and UNIX-like operating systems which allocates the computers resources to other programs. The Linux Kernel would streamline the already existing operating system (GNU) creating a more efficient and reliable structure, this system would be called GNU/Linux, and though much to the frustration of Richard Stallman, the GNU has been dropped from the common nomenclature and is now most commonly referred to as Linux. In 1991 Linus Torvalds, while studying at the University of Helsinki, created the Linux kernel in his spare time and since its release on September 17, 1991, this kernel and Torvalds direction of its distribution has created a boon for Open Source software within the marketplace.

Torvalds not only developed the last piece of the GNU puzzle, he also utilized the open source community better than any one had previously. Within a month of its first release, Torvalds, with the aid of hundreds of community programmers representing thousands of programming man hours, released the second edition of Linux, demonstrating how effective a decentralized programming network could be. Over the next few years Torvalds refined this process by creating a hierarchical network of programmers constantly working (under Torvalds umbrella, a type of quazi-control and direction) on the betterment of Linux and Linux programs. From the start, Torvalds would utilize only the best suggested refinements to the code. With the success of the GNU/Linux operating system, more and more users including large corporations began to take notice and show interest in the possibilities of how this new program and model of development could be utilized, but the stigma of “Free” Software made many in the private sector uneasy.

With the publication of Netscape’s source code into free software in 1998, a group of GNU/Linux and Free Software users including Todd Anderson, Chris Peterson, and Eric Raymond and others, saw a need to change the public perception of the Free Software Movement and created the Open Source Initiative. This group created a more business friendly model for promotion and licensing. Under the direction of the Open Source Initiative, a definition of what Open Source should be was created and promoted as an attempt to dispel private and public user’s misconceptions of what the “free” in Free Software stood for.

The Open Source Definition

*1. **Free Redistribution**. The license may not restrict any party from selling or giving away the software as a component or an aggregate software distribution containing several programs from several sources. The license may not require a royalty or other fee for such sale.*

*2. **Source Code**. The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost -- preferably, downloading via the Internet without charge. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a preprocessor or translator are not allowed.*

*3. **Derived Works**. The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software*

*4. **Integrity of The Author's Source Code**. The license may restrict source-code from being distributed in modified form only if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.*

*5. **No Discrimination Against Persons or Groups**. The license must not discriminate against any person or group of persons.*

*6. **No Discrimination Against Fields of Endeavor**. The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.*

*7. **Distribution of License**. The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.*

*8. **License Must Not Be Specific to a** Product. The rights attached to the program must not depend on the program's being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution.*

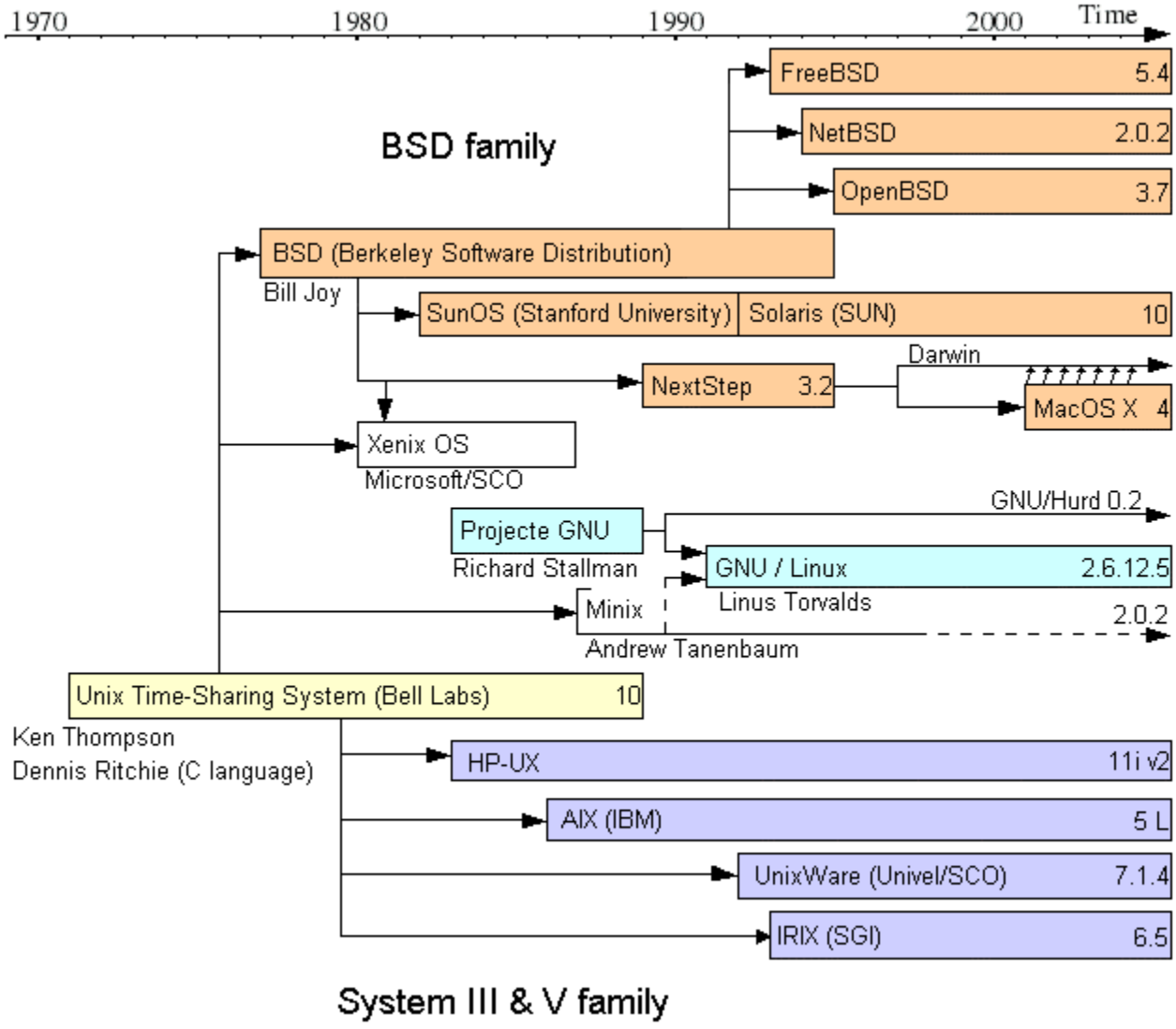
*9. **License Must Not Contaminate Other Software**. The license must not place restrictions on other software that is distributed along with the licensed software.*

For example, the license must not insist that all other programs distributed on the same medium must be open-source software

(opensource.org)

While these terms are written in the form of a social contract they have for the most part been accepted and adhered to by the now re-named "Open Source" community. This is largely due to the influence of many of the key figures within the community such as Torvalds and Eric Raymond (president of the Open Source Initiative and author of The Cathedral and the Bazaar). Although for many Open Source enthusiasts the underling philosophy created by Stallman of information sharing and the free trade of ideas remain the cornerstone of this movement.

Time line of UNIX Family Tree



Development and Distribution

In his essay The Cathedral and the Bazaar, Eric Raymond describes the Linux development model by differentiating between Linux and more traditional programming as two sides of a drastically different building scheme.

> *Linus Torvalds's style of development—release early and often, delegate everything you can, be open to the point of promiscuity—came as a surprise. No quiet, reverent cathedral-building here—rather, the Linux community seemed to resemble a great babbling bazaar of differing agendas and approaches … out of which a coherent and stable system could seemingly emerge only by a succession of miracles.* (cathedral)

This analogy, likening traditional programming to the strictness and structure of cathedral building, illustrates in a dramatic fashion the way in which Torvalds utilizes the seemingly chaotic nature of his development strategies to gain advantage over proprietary systems. One of the most important ways in which Open Source programs and operating systems are far superior to many of their proprietary counterparts is in the interactions between the software and other programmers. For most proprietary programs and operating systems this interaction has become a "can you break this" approach. Because of this, large teams of programmers under strict constraints releasing products in the market hope that they were able to isolate all of the vulnerabilities within the program. While for Open Source programs and operating systems, users not only find and fix problems in software, they find new and interesting ways to improve on it. This type of interchange has made Open Source programs more readily adaptable to individuals looking for ways in which to adjust software to meet certain criteria unique to their own needs.

This however has been one of the main perceived drawbacks to Open Source software, most programmers and subsequent users of Open Source traditionally have more computer skills than the average user, raising complaints that Open Source programs were not user friendly for the average user. While for the most part the Linux community has taken great strides in fixing this problem on a programming level, they have done little to dispel this misconception to the common user. This decentralized form of development in which often lone programmers work on sections of programs or even more recent trends of teams of programmers hired by such companies as IBM and HP, has allowed Linux the opportunity to create a library of programs and operating systems that would have cost in the billions of dollars in the traditional market. Only within the last few years has Torvalds relinquished some of his power to others capable of furthering the Linux agenda. Linux now also has a Board of Directors guiding it to more consistent goals, allowing Torvalds to concentrate on the quality of development.

Although Linux continues to gain market share on rival Microsoft, Microsoft has several inherent advantages to this decentralized model of development. First is market

place reputation; for most users there is an unwritten contractual understanding that Microsoft will always fix problems as they arise, but with Linux there is only a veiled presumption that this will take place, and when it does, the release dates are often times sporadic. Should a user find that a problem affects too few other users for the community to take notice, he might be left to his own devices. The second inherent advantage is this: Microsoft has shown repeatedly that the release of new products and upgrades arrive on a consistent schedule, where as Linux users often are at the mercy of an unknown entity to release a needed product or upgrade.

Despite these hindrances many large corporations are entering the scene either as backers or users of Linux-based products. Companies such as IBM, Novell, Intel and HP have not only brought with them market credibility, but also leagues of professional programmers that strengthen Linux's ability to compete with Microsoft. In a Businessweek article written in January, 2005, IBM is reported as having 600 programmers now working exclusively on Linux, up from two in 1999. Due in large part to these changes, Linux's market share from 1997 to 2003 in servers rose from 6.8% to 24%, and this number is expected to reach 33% by 2007. (Businessweek) With IBM and HP now pre-loading PC's and servers with Linux many more common users may find more and more reasons to switch from Microsoft to Open Source systems.

While for the time being Microsoft is secure in its place at the table, for the first time in many years they seem to have a competitor nipping at their heels; only time will dictate the extent to which Linux will challenge Microsoft's market dominance. Many experts postulate that the recent upward trend of Linux's market share is due to the drop in UNIX share and that once this trend equalizes, Linux will flatten out. Others still have an enthusiastic optimism about Linux's possibilities and overall outcome. Regardless of Linux and Open Source software's marketability, much like the internet and World Wide Web before it, the way in which it has changed the marketplace of ideas will be its most lasting contribution.

Inherent Security Advantages

There are several inherent security advantages to the open source software development model. They are discussed below.

Security Through Disclosure

Open source software is distributed with its sources and hence it is analyzed and reviewed by a large number of individuals in the open source community and by developers who are trying to modify the code for their own use. This level of peer review cannot be achieved by companies developing proprietary software. As a result of the large number of reviews several vulnerabilities are found and fixed before they get exploited "to many eyes, all bugs are shallow"[1]. There are several examples of security

[1] ES Raymond, \The Cathedral and the Bazaar", 1998, at http://tuxedo.org/~esr/writings/cathedral-bazaar/

flaws being discovered by code reviews in the open source, for instance security vulnerabilities discovered in CVS[2].

Open source also provides accountability against developers who may place malicious code like Trojans and other backdoors in the software. For example Borland's InterBase database had a backdoor that allowed any local or remote user to take over the system as root. This hole remained undetected for several years until Borland made the sources public and the flaw was discovered and reported with in six months of the release of the source code. Similarly in January 1999 a Trojan Horse version of TCP Wrapper was placed on a popular website. Due to the availability of source code, the backdoor was quickly identified and removed.

Software Complexity

Open source software is usually less complex, more modular and readable than closed systems. For example Windows is estimated to have between 40 and 60 million lines of code, as compared to Linux with around 5 million. The smaller complexity of the open source makes it easier to comprehend, review and fix.

Communal interest in Security of Open Source Software (OSS)

Every user and developer of open source software has a vested interest in the security of the open source systems. Hence community plays a vital role in making OSS more secure. The open source community is more receptive to vulnerabilities reported in their software as opposed to proprietary software vendors and therefore more nimble in their response. For example a serious vulnerability found recently in Sony's DRM rootkit took a couple of weeks and a lot of outcry from the customers before the vendor finally acknowledged and decided to fix to the issue[3] .

On the other hand the good will of the open source community and the free exchange of source code allows for quick discovery of flaws, responsible disclosure and fast development and deployment of security patches. This communal effect is unique to OSS.

Support -- you may fix it yourself!

Lack of support is often cited as one of the biggest hurdle in adopting open source software. However, this is not entirely true. Since the source code is available, anyone can provide support for the software.

When a security flaw is discovered in a proprietary software system, the customers have to wait till the vendor develops and provides a patch. The time taken between the discovery of an exploit and the patch being available is notoriously long. To make the

---

[2] SuSE: New cvs packages fix remote command execution
http://freshmeat.net/articles/view/1216/

[3] Real Story of the Rogue Rootkit
http://www.wired.com/news/privacy/0,1848,69601,00.html

situation even worse, a lot of the time patches are released prematurely with insufficient peer reviews and testing and end up causing more vulnerabilities. For example Sony's patch to the recent rootkit flaw causes further vulnerabilities[4].

In the case of OSS, anyone in the open source community may fix the problem and since the code of the patch is also open source, it gets peer reviewed and tested rigorously hence the response time is shorter and the quality of the patch is better.

Full Disclosure of Bugs and holes

Users of proprietary software systems have to rely on the honesty and diligence of the software vendor to have shipped a high quality and secure product. However not all software vendors are as diligent as they need to be. Some vendors ship software with known vulnerabilities largely due to financial reasons for instance to release the product on time, cost of fix etc. Some vendors also take comfort in the complexity of the exploit and think that no one will discover it. Most software vendors feel very uncomfortable about the publicity of security flaws and argue that this helps the attackers. The open source community has taken a different approach: security through disclosure. The open source community is committed to sharing of knowledge about security flaws and possible exploits through newsgroups and mailing lists. The (responsible) public disclosure of security flaws has benefited the overall security of the OSS.

Behavior of developers

Most discussions about software quality revolve around hours of testing and mean time before failure of the system. The value of testing is well understood, however, we must acknowledge that testing does not guarantee absence of flaws. And the best way to make quality software is to build quality/security into the system, ie, improve the way the software is designed and developed. Research in human behavior and psychology has shown that people demonstrate their best behavior and perform better when they are under observation[5]. Given the natural human behavior, it is more likely that a developer will write better code that is well designed, well structured/modularized, readable and maintainable. The 'fear' of being scrutinized publicly makes developers write better more secure code to start with.

Diversity

The diversity of the open source community also brings an interesting dimension to the quality of the software. Research has shown that individual preferences and skills of individuals have a profound affect on testing. A piece of software may have been tested extensively by one person to a point where it seems to be reliable but when given to

[4] Sony DRM infection removal vulnerability uncovered
http://www.theinquirer.net/?article=27714
[5] The Immediate Effects of Being Observed
http://www.obmnetwork.com/resources/articles/ABA2004/Austin_Observed.ppt#272,15,Method

another tester, quickly reveals flaws[6]. Also test suites developed by testers employed by proprietary software vendors tend to become stale after the initial test development phase and stop catching new bugs. On the other hand open source community has unparalleled diversity and 'work force' of testers in it's members. Due to this reason open source community is likely to find more bugs sooner.

OSS Security Tools

One of the big advantages of using open source is the availability of open source security tools. These tools are freely available and can be used by anyone. This makes open source a very attractive option for small businesses and individuals who cannot afford expensive support and service contracts.

Code Audit

Open source also has the advantage that you can audit the code. As mentioned earlier proprietary software systems have the possibility of undiscovered backdoors. For people and agencies who are interested in deploying secure systems open source provides the unparalleled opportunity to audit the code (or pay someone to do it) to satisfy their requirements.

Inherit Security Disadvantages

The following section discusses inherent security disadvantages of open source software and follows with a discussion of how the open source community responds to security problems.

The Downside of the "Many Eyes" Approach

One obvious disadvantage of open source software is that the source code is available for examination to everyone, including attackers. It is true that source code is not required to understand the workings of a program or to find vulnerabilities, which may be accomplished through reverse engineering the machine code. However, it is more time consuming and requires a greater degree of skill to understand machine code. Security through obscurity is fragile, but it can make discovery of a vulnerability and development of a successful attack more difficult. This is one layer of defense that open source software lacks.

Related to this point is a refutation of the "many eyes" argument. In the security context, the "many eyes" argument is taken to mean that with the source code of an open source program available to everyone, many people will review the code and quickly find and fix any security problems. However, the typical open source programmer is seeking to *modify* some aspect of the software to add a new feature or fix a bug, not to perform a comprehensive security review, which is both time consuming and difficult. So the availability of an open source program will likely benefit attackers more and defenders less than is commonly perceived within the open source community.

---

[6] RM Brady, RJ Anderson, RC Ball, \Murphy's law, the fitness of evolving species,and the limits of software reliability", Cambridge University Computer LaboratoryTechnical Report no. 471 (September 1999), available at http://www.cl.cam.ac.uk/ftp/users/rja14/babtr.pdf

Accountability

Another disadvantage of open source software with respect to closed source software is the relative lack of accountability. Anyone can contribute code to an open source project (though it may be reviewed and not necessarily accepted in some cases). An attacker could attempt to add subtly malicious code to an otherwise benign open source program to implement a Trojan horse attack against users of that program. As noted above, such an attempt was made against the TCP Wrappers program in 1999. The attempt was discovered quickly; however, a more cleverly disguised attack may have evaded detection for longer.

Although counterexamples exist, this is a threat that seems much less likely with closed source software produced by a company. An employee who launches such an attack is endangering his livelihood. Additionally, his identity is known by his employer so he may be more easily prosecuted than a relatively anonymous open source developer contributing code over the Internet.

Development Style and Discipline

A final disadvantage of open source software is the relatively relaxed, unstructured way in which software development occurs in that community. Creating secure software is a difficult, complex task. Companies like Microsoft have discovered that disciplined engineering processes are necessary to achieve this. Open source development often lacks such disciplined processes. Closed source software is often created in a corporate environment where it is easier to enforce such processes and discipline.

Open Source Responses to Security Problems

The responsibility for responding to a security flaw in open source software typically falls on the community that creates that software. A patch for the flaw is typically created, and then distributed in the same manner that the program is distributed (e.g. downloading from an FTP server). An advisory for the flaw may be posted at various security mailing lists or sites, such as bugtraq.

There is no uniform policy across all open source projects about how an open source vulnerability should be disclosed. One major open source project, Mozilla, has established a policy[7] that attempts to reconcile two competing interests: (a) the desire to be as transparent as possible and disclose vulnerabilities publicly and (b) deny attackers a short-term advantage in exploiting the problem. Mozilla security bugs may be flagged as "security sensitive" in their bug database, which controls access to the bug. Certain Mozilla project members as well as the bug reporter have access to the bug. The bug reporter may decide to make the bug public at any time.

Recourse for cyber attack victims is limited. As with closed source software, license terms often restrict whether victims may sue the makers of the software. It is reasonable to assume that major open source projects such as Linux or Mozilla will be fairly responsive to security problems and will fix them. Projects of this size have large user

[7] http://www.mozilla.org/projects/security/security-bugs-policy.html

bases they need to support and also large numbers of programmers to fix the problem. Additionally, they are backed by major corporations (such as Red Hat, IBM or AOL) who have interests in seeing such problems fixed. Smaller, more obscure open source projects may be less well-maintained, in which case security problems may receive less attention.

Comparison of Security in Open Source Versus Other Products

Case Study 1: Linux Versus Windows

In this section we compare Windows Server 2003 Datacenter Edition and Red Hat Advanced Server with respect to security/vulnerability and how that relates to open source versus proprietary software.

Both Linux and Windows are very complicated software packages. Windows is estimated to have between 40 and 60 million lines of code and Linux has approximately 5 million. Complicated systems are much more difficult to build flawlessly. Windows is about 10 times bigger than Linux which would makes it much more difficult to secure.

Windows also has a much bigger deployment base compared to Linux. This makes it a more attractive and a more available target for attackers. Furthermore, a Windows exploit has a more damaging effect to the industry than a Linux exploit.

Issue Discovery

We used trends gathered by Secunia to compare vulnerability patterns between Windows[8] and Linux[9].

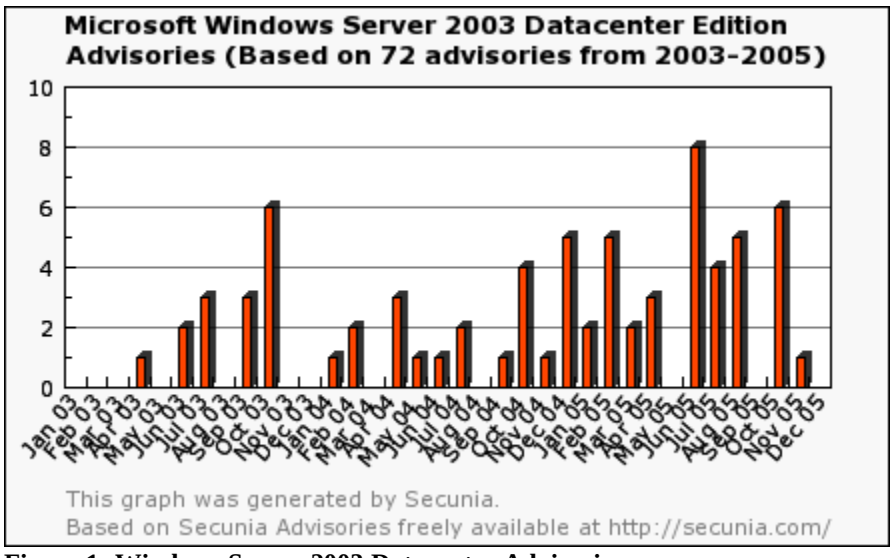


Figure 1: Windows Server 2003 Datacenter Advisories

[8] http://secunia.com/product/1175
[9] http://secunia.com/product/2534

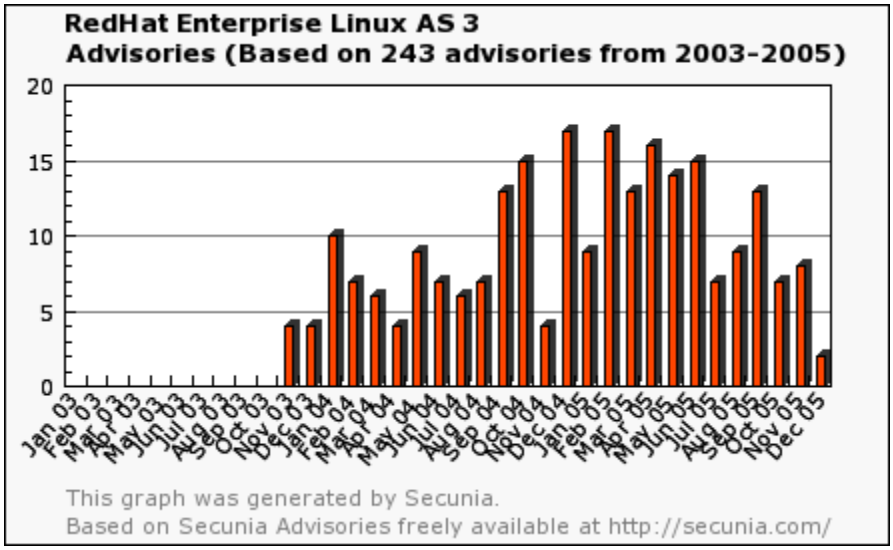


Figure 2: RedHat Enterprise Linux AS 3 Advisories

As we can see from Figure 1 and Figure 2, the amount of advisories issued against Linux is over three times as many compared to Windows (243 versus 72). Interestingly, the chart shows that the issues raised against Linux climbed rapidly after release, reached a peak and then started to fall. Almost a year elapsed from the initial advisory to the peak volume of advisories. That makes sense, since issues begin to surface after the product is adopted by consumers.

Surprisingly, the Windows advisories did not follow such a pattern. The trend shows that the Linux issues were found rapidly in high volume whereas the Windows issues were found in a somewhat erratic fashion and do not show signs of significant decline. This could be due to various reasons. Firstly, a slower adoption of the new Windows version would increase the amount of time it takes for issues to surface. Secondly, and more interestingly, since Windows is not open source, it is more difficult to find the issues. Without access to the source code, finding vulnerabilities can be a very time consuming trial-and-error process.

Fixing Issues

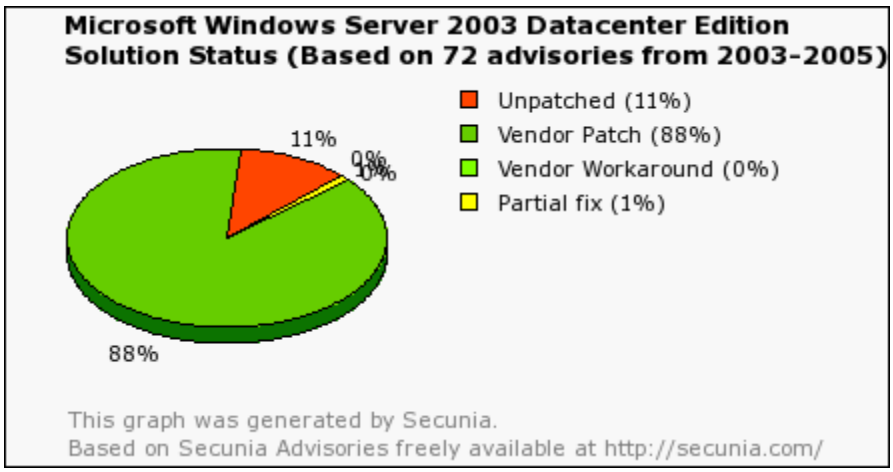


Figure 3: Breakdown of solutions applied to vulnerable Windows machines

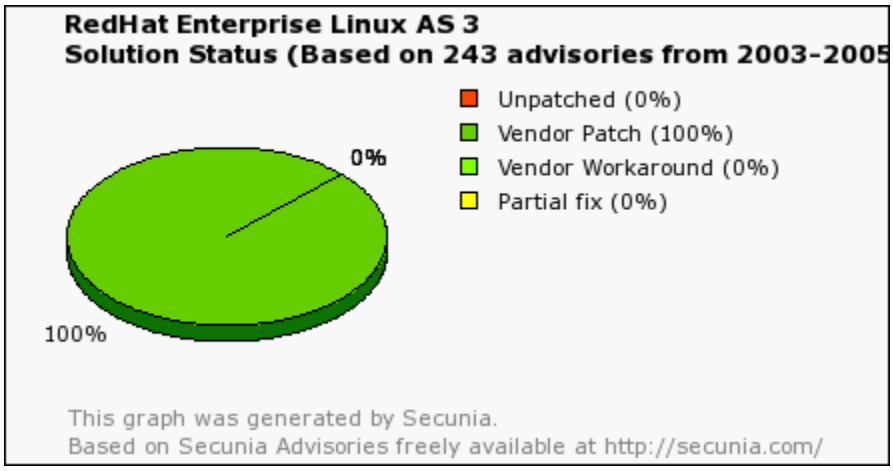


Figure 4: Breakdown of solutions applied to vulnerable Linux machines

From Figure 3 and Figure 4, we see that all the Linux patches were applied whereas only 88% of vulnerable Windows machines were patched. This is surprising considering the amount of energy and infrastructure Microsoft devotes to providing customers with automatic downloads and installs of patches.

The Microsoft security bulletins appear to be better than those published by Red Hat[10]. The Microsoft bulletins describe mitigations, workarounds, and patch installation walkthroughs. The RedHat bulletins, on the other hand, are very succinct and mostly just describe the vulnerability.

This demonstrates that Microsoft must do even more if it wants all its customers to have patches machines. The success rate of Linux patches compared to Windows can be explained due to the low-impact that Linux patches tend to have[11]. Linux patches can often be applied without seriously rebooting or bringing down the machine whereas Windows patches usually require reboots or some cause some significant disturbance to the service being provided by the machine.

[10] http://www.sisecure.com/pdf/windows_linux_final_study.pdf
[11] http://www.sisecure.com/pdf/windows_linux_final_study.pdf

Issue Severity

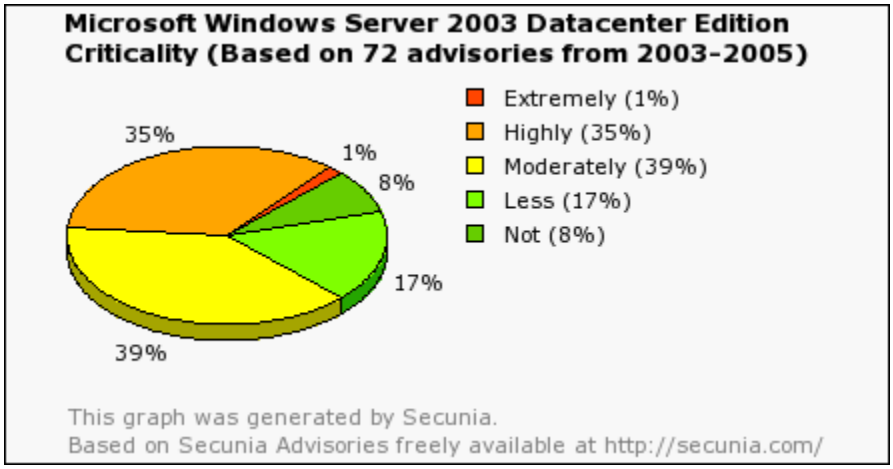


Figure 5: Severity of issues found in Windows

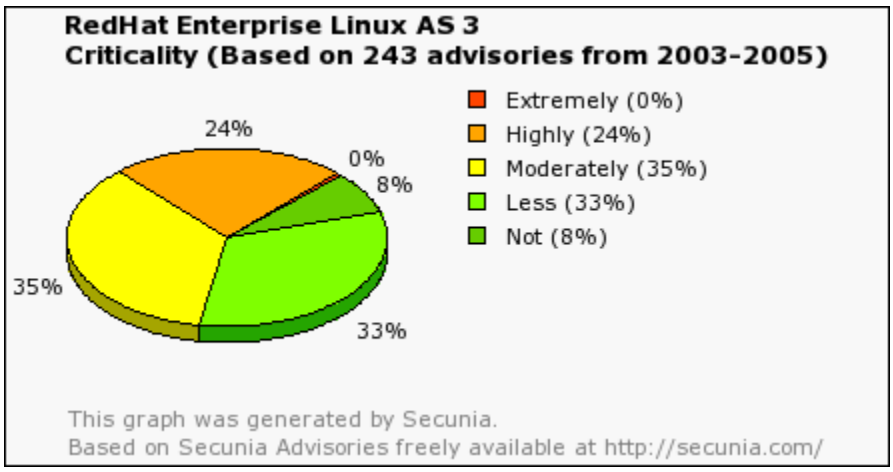


Figure 6: Severity of issues found in Linux

From Figure 5 we see that Windows has 36% issues that are extremely or highly critical, 39% moderately critical, and 25% less or not critical. However, Linux has 24% extremely or highly critical, 35% moderately critical, and 41% less or not critical. Albeit, Windows has a lot less issues, it has a higher percentage of critical issues.

Exploit Environment

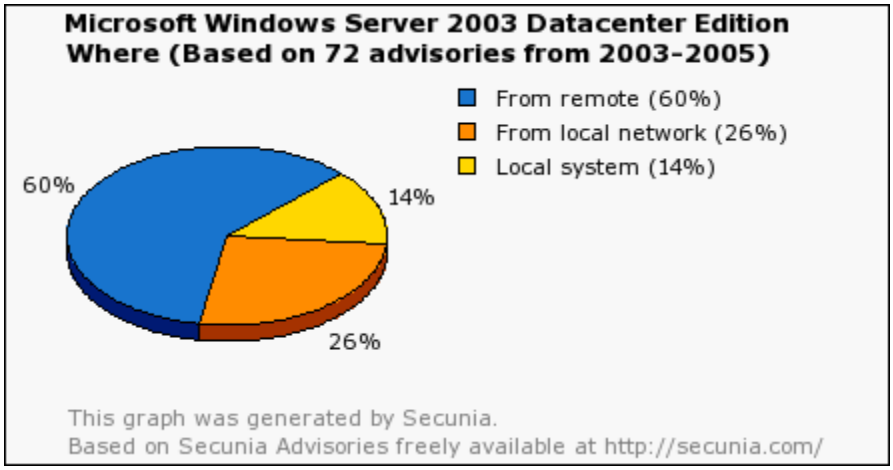


Figure 7: Locations from where Windows issues can be exploited

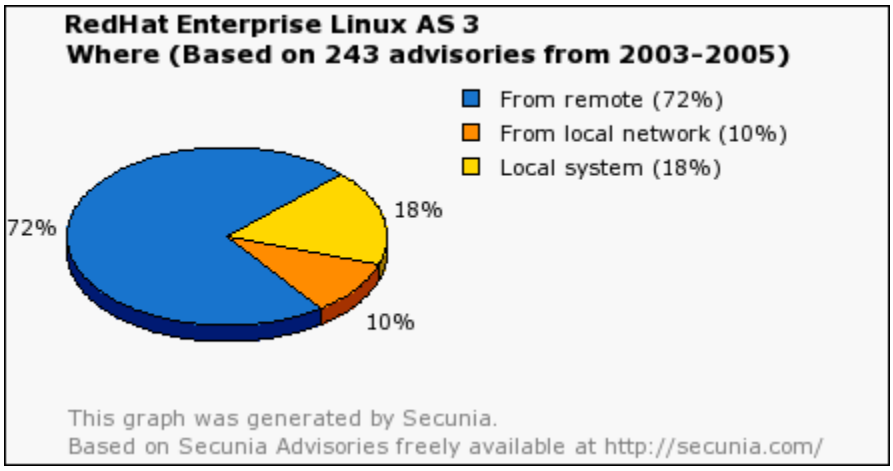


Figure 8: Locations from where Linux issues can be exploited

Figure 7 and Figure 8 shows whether the vulnerabilities can be exploited by using another machine on the network or whether they require physical access to the machine. We see that both Windows and Linux are close (86% versus 82%) in regard to the percentage of vulnerabilities that can be exploited without being locally logged into the machine. However, Linux is more susceptible to an exploit being launched from a remote network. That is, a higher percentage of Linux vulnerabilities can be exploited from the Internet.

Summary

In summary, although there are three times as many advisories posted against Linux, we do not see any direct evidence that open source is the reason for insecure software or closed source is the reason for securer software. The security of software really depends on the software development process and the engineering practices followed the development team. Considering security throughout every step of the development life cycle (e.g. specification, design, implementation, testing) is critical to building secure software. The low defect count seen in Windows is a result of the Microsoft being proactive about security in its software development practices.

Open source does demonstrate an advantage with regard to vulnerability discovery. Vulnerabilities in Linux were found rapidly, and after reaching a peak, progress to a downward decline. Vulnerability discovery in Windows is more erratic and does not demonstrate a clear downward trend. As a result of being open source, the Linux source code can be studied by everyone to find the issues. However, with Windows attackers and security experts are left with a more time consuming trial-and-error approach.

We also see that the disturbance caused by having to patch the machines is expensive enough for machine owners to take the risk of running vulnerable machines. Linux machines can be patched without any service outage whereas Windows machines often have to be rebooted. As a result, Linux owners are more likely to patch their machines and Windows owners are not.

Case Study 2: MySQL Versus SQL Server and Oracle

We conducted a study of the vulnerabilities reported in the popular open source database: MySQL and Microsoft's SQL Server 2000. Both products have been in the market for about the same time. We used SecurityFocus.com's vulnerability database for our study. We compared the vulnerabilities found in the two products on dimensions like defect class, exploit class etc. The observations are summarized in the table below:

	MySQL	**SQL Server 2000**
Total vulnerabilities reported since 2000 *	53	40
Reported in 2000	3	8
Reported in 2001	4	4
Reported in 2002	10	22
Reported in 2003	8	5
Reported in 2004	15	0
Reported in 2005	14	1
Remote	36	25
Local	16	14
Defect: Boundary condition error/buffer overflow	17	22
Defect: Design Error	13	6
Defect: Input validation error	6	2
Defect: Access validation error	5	3
Defect: Failure to Handle Exceptional Conditions	5	4
Defect: Configuration Error	4	1
Defect: Environment Error	1	0
Result: Denial of Service (DOS)	10	6
Result: Execute arbitrary code	19	22
Result: Information disclosure	5	2
Result: Elevation of privileges	6	8
Result: Script injection	2	2
Result: Admin cannot detect attack	1	0
Result: Account/session hijacking	6	3
Result: Unauthorized remote access	5	0
Result: Disclosure of credentials	5	2
Result: Data/file corruption	3	0

Exploit available **	19	20
Exploit not available **	35	20

It is interesting to note that a comparable number of vulnerabilities have been reported for both MySQL and SQL Server 2000. It is also interesting to note that the most common defect by far was buffer overflow in both products. MySQL has, however, relatively larger proportion on design flaws.

The most common exploits reported in both products were denial of service (DOS) attacks and execution of arbitrary code using buffer overflow exploits.

One big difference between the two products is the trend of the new vulnerability reports. Where MySQL seems to have a more or less steady flow of vulnerability reports, SQL Sever 2000 has seen a sharp decline in the past couple of years. This may be attributed to a couple of factors:

1. Microsoft did not release new versions of SQL while several new versions of MySQL has been released during this time.
2. Microsoft did a security push in 2002/2003 and this may be an indication that there are not many easy to find exploits left.

* The number is based on the total number of reports in the SecurityFocus.com database. Some reports had multiple vulnerabilities but they have been counted as one for the purpose of this study.
** This information is based on SecurityFocus.com database. It is possible that exploits may be available on the web but not reported on the website.

Case Study 3: IE Versus Firefox

Introduction

This case study will compare Internet Explorer 6 (IE) and Mozilla Firefox based on their security flaws and features, how their code bases are managed and the security incident statistics. Internet Explorer is a proprietary product developed by Microsoft and Firefox is an open-source product managed by the Mozilla Foundation. This case study will drill down more on how open source techniques can affect the effectiveness of dealing with security bugs and design. This case study will not compare each browser on features that are not directly related to security such as tabbed pages and RSS feed support.

Background

Both IE and Firefox are web browsers designed to browse the Web. IE 1.0 was first released in 1995. It underwent several revisions until version 6 which was released in 2001. Firefox was first released under the codename ‘Phoenix’ which was made public in 2002 and is based on the Mozilla Foundation’s code base. After several name changes, the latest version available is Firefox 1.0.2. According to PCWorld.com, IE currently has a market share of 94%, whereas the market share for Mozilla browsers such as Firefox and Netscape is now at approximately 4% of all users. IE and Firefox publish their latest security flaws on a regular basis. Based on a review of recent security bulletins, both

browsers have exposed vulnerabilities of being able to execute code remotely. Firefox allowed this through Linux and IE allowed this through the Windows operating system. Statistics based on how long it takes to fix these bugs is not widely available for both browsers.

Reporting Security Bugs

Mozilla regularly fixes security bugs without informing its user base. IE publishes security bulletins for almost every bug found as this is company policy. Firefox does not also have an automatic way for users to update their installs, whereas IE uses Windows Update to automatically update a user's install. In addition to this, IE releases security updates on the second Tuesday of every month so that enterprise and consumers can plan for these updates. Firefox's inability to automatically update its customer's installs potentially leaves a large number of its customers with many different versions of unpatched Firefox installs.

Firefox allows any users to report security bugs whereas IE allows user to report issues but does not give them access to its internal bug tracking system to follow the progress of the fix. Firefox has also allowed for it's users to be rewarded monetarily for security bugs that they find. As this is a recent initiative by the Mozilla foundation, there are no meaningful statistics available yet on whether this has increased the number of bugs found. Microsoft Corporation has teams of penetration testers who attack products like IE so the reliance on external bug reporters is less than Firefox.

Security Features

Both Firefox and IE contain a plethora of security features with both claiming to have features unique to each. There are still a few similar security features common to both such as Pop-up blocking and the ability to purge personal data such as browsing history, cookies, webform entries and passwords.

The key difference between IE and Firefox is that Firefox is not completely integrated with Windows so that viruses attacking Windows will have minimal impact on Firefox especially since Firefox is not closely integrated with the Windows file system and network stack. Firefox also has no support for VBScript and ActiveX, both which are sources for many security holes. Firefox does not use Microsoft's Java Virtual Machine, which has a history of more flaws than other Java VMs.

On the other hand, IE contains additional security features which do not exist in Firefox. For example, IE has the concept of zones which allow the user to put trusted sites in 'Trusted' zones. This partition allows for trusted sites to be handled differently from untrusted sites. For example a trusted site can be allowed to download ActiveX controls without prompting. IE also has the ability to selectively block ActiveX usage which allows the user to be prompted if an ActiveX control is to be downloaded. By being closely integrated with the Windows operating system, users of IE automatically get many of the Windows XP security features such as Windows automatic updates and the Security Center containing tools to detect security vulnerabilities.

Checking in Code

IE and Firefox have very different processes for checking in code. Mozilla Firefox does not necessarily have security reviews done on code before it is checked in. There are assigned 'module' owners who are available to review code; however, this is not mandatory for a checkin to be made. The fact that there are no consistent code reviews opens the door to more potential security bugs especially when there are no code reviewers dedicated to detecting security issues.

By contrast, IE has dedicated development and testing teams that have a strict process whereby code is peer-reviewed and tested before any checkins are made. Also, the Windows Division has a Secure Windows Initiative which is a team of security experts within Microsoft that review all components checked into the Windows code base.

Firefox has a security policy that is subject to change and has changed based on the opinions and votes of its user community. Firefox also has a security module owner who is responsible for reviewing code only when security fixes are made. This is in contrast to IE which does regular security reviews of all code. However, code that gets checked into the Firefox code base potentially has more public exposure as there could be many more people involved in a code review than the IE team which is usually of a fixed size. The IE team will have a fixed number of people working on the product at one time and given the environment at Microsoft it is still possible that security may take on a secondary role when the IE team is in 'crunch' mode and must deliver on a release by a specific date.

The main advantage IE has over Firefox in terms of dealing with security bugs is the fact that there is a dedicated team that will concentrate on security issues and performing penetration testing on the product. However, Firefox has a lot more exposure given the size of the developer community who contribute to the code base, which is sizeable especially since Firefox is open source.

Statistics

One way of determining how secure a product is by analyzing the number and type of security flaws found in the product in any given time period. Secunia[12] is a renowned security research company that monitors security flaws in thousands of products. The following sections will discuss Secunia's finding on the security flaws found in both IE and Firefox over the last twelve months as Firefox was used extensively only recently based on usage data available on its website. Data from the last three years related to IE will also be discussed to illustrate any trends.

Internet Explorer http://secunia.com/product/11/

[12] http://www.secunia.com

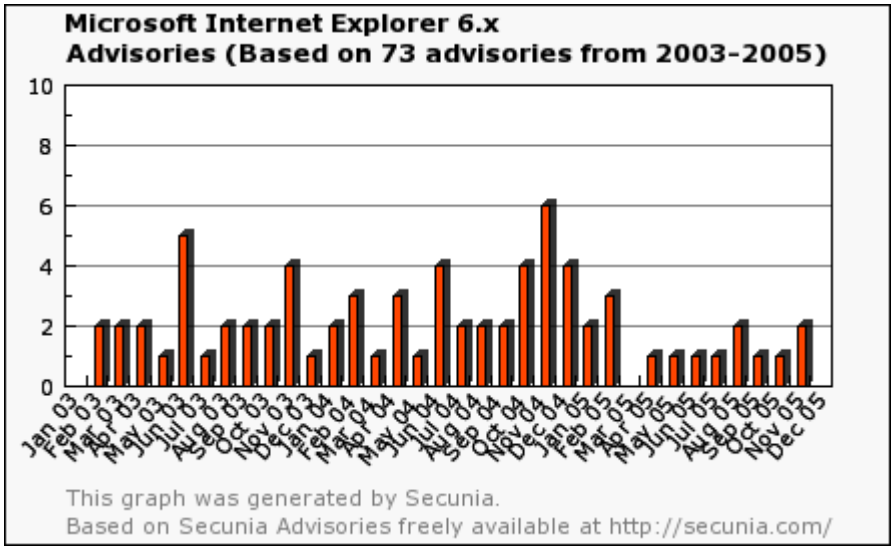


Fig 1-1 : IE 6 Advisories from 2003-2005

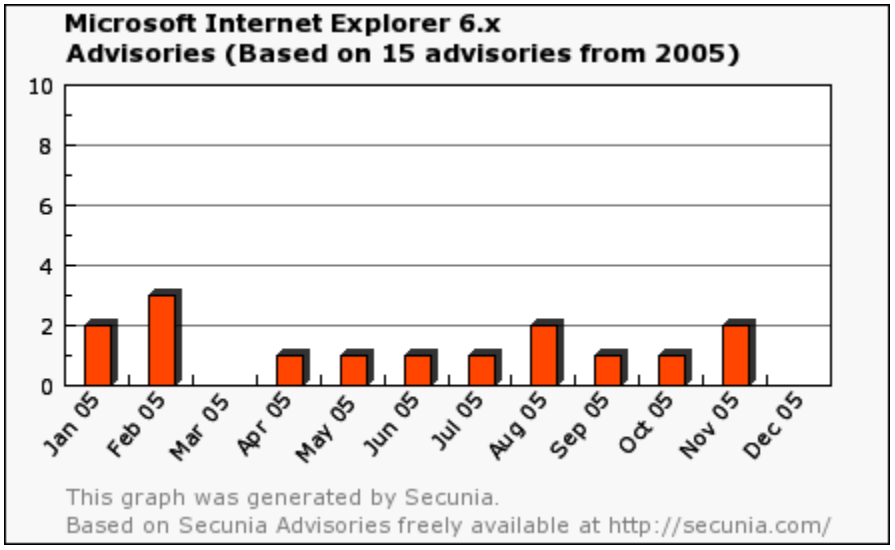


Fig 1-2 : IE 6 Advisories in 2005

Mozilla Firefox http://secunia.com/product/4227/

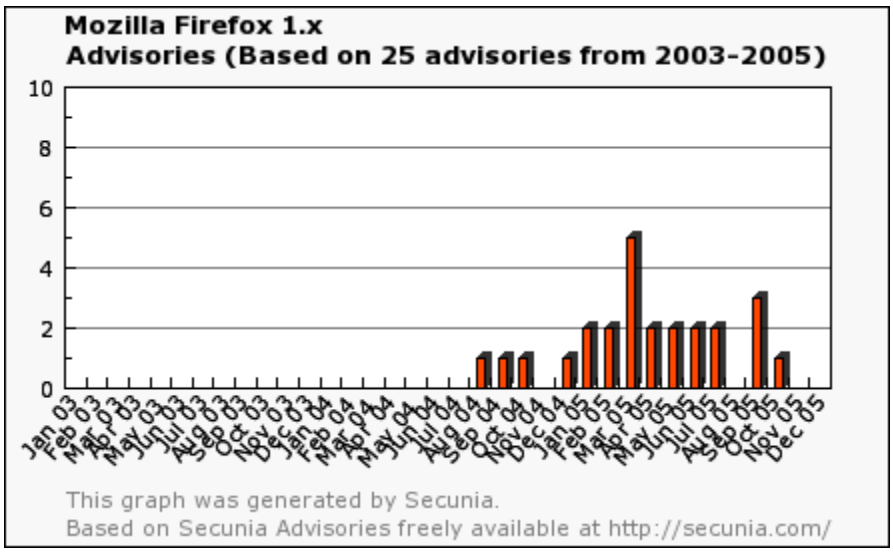


Fig 2-1: Mozilla Firefox Advisories from 2003-2005

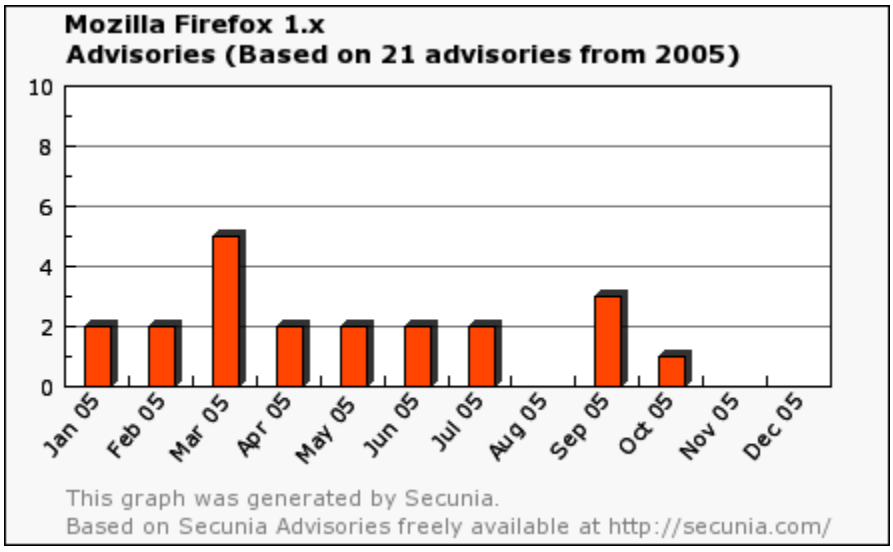


Fig 2-2: Mozilla Firefox Advisories in 2005

Fig 1-1 shows that IE has been long plagued with many security advisories during the last three years. However, the number of security advisories was reduced substantially after the introduction of IE 6 which was released with Windows XP SP2. Fig 2-1 also shows that the number of advisories increased substantially when Mozilla Firefox started to gain popularity which is indicative of its increased market share of the browser market.

Fig 1-2 also indicates that there were only 15 security advisories issued this year so far, as opposed to Fig 2-1 which shows that there were 21 advisories reported for Firefox during 2005. However, according to Secunia, out of the current 15 issues reported for IE this year, 47% of these have been unpatched as opposed to Firefox which has 0% security issues which are unpatched. Looking at these statistics in more detail uncovers the fact that 95% of the security issues for Firefox were patched by the vendor, whereas only 40%

of the issues for IE were patched by the vendor. This is indicative of the open-source nature of Firefox where vendors who use Firefox invest their own resources to fix problems. Microsoft vendors have limited access to the IE source code so the percentage of vendor patches is smaller.

Secunia also reports on the criticality of reported vulnerabilities based on how a malicious user could gain root access to the system or cause denial-of-service (DoS) attacks, for example. The statistics show that 5% of Firefox vulnerabilities are 'Extremely Critical' as opposed to 13% of IE vulnerabilities. However, given that these are percentages, this account for approximately 2 security issues for Firefox and 2 issues for IE.

The statistics above cannot conclusively determine how secure one product is as opposed to each other, especially given the fact that Firefox has had a smaller market share and does not have the ten year history that IE has. However, they do indicate that as Firefox has become popular, the number of reported vulnerabilities has increased. Also, it appears that IE has also started a downward trend towards less vulnerabilities reported which may be attributed to its recent security push which started during the release of Windows XP SP2.

Conclusions

Both Firefox and IE have been the target of recent media scrutiny whenever new security bugs are found in their products. Firefox has started to attract a substantial number of users and has taken some of the IE's market share. The fact that Firefox is open-source and is not integrated with the Windows operating system has led many to believe that it is less vulnerable to security attacks. However, this case study has shown that in the past year it has had more vulnerabilities reported than in IE. The way code is checked in, security bugs are reported and fixes are reviewed may have contributed to this rise in security issues. Firefox also lacks automatic updates which is not necessarily a side-effect of open source but may be due to the fact that Firefox cannot piggyback on an operating system such as Windows. For an open-source system, the infrastructure required to allow automatic updated may not be feasible or even possible to create. Internet Explorer, which is not open-source, is a mature codebase with dedicated development and test teams. However, it is also subject to security flaws and vulnerabilities, some of which are the result of it being too closely integrated with the Windows operating system.

Given all the above factors, it can be concluded that an open source product such as Firefox may lend itself to more than or an equal number of security flaws as compared to a proprietary product like IE. However, given the complexity of the Internet space and operating system dependencies, open-source products are not inherently insecure especially given the fact that IE, with all its stringent checkin policies and security reviews, is still subject to many security flaws even after 10 years of existence. It is possible that over time Firefox may take even more market share from IE especially if it focuses on refining its checkin policies and security reviews, and perhaps incorporate infrastructure such as automatic updates and adding more security features that are already being used in the Windows operating system.

Links:
http://www.mozilla.org/security/announce/mfsa2005-59.html
http://www.mozilla.org/projects/security/known-vulnerabilities.html
http://www.mozilla.org/projects/security/security-bugs-policy.html
http://www.microsoft.com/technet/security/Bulletin/MS05-052.mspx
http://www.linuxpipeline.com/165600315

Overall Conclusions

We found significant advantages and disadvantages of open source software from a security standpoint. The advantages and disadvantages center around two things: the open-ness of the code, and the nature of the open source community. By the "many eyes" argument, the open-ness of the code means many defenders will examine the code and find and fix security issues. The corollary to this, of course, is that attackers can also examine this code to find and exploit vulnerabilities more easily. The open source community is large, diverse, and concerned about writing secure code that becomes public. This same community, however, is not as accountable or structured as engineers working for a company.

From our three case studies, we found that open source products have reported more vulnerabilities than comparable closed-source products. Open source products do not seem to have any significant advantage. Additionally, we found that the numbers of IE and SQL Server vulnerabilities have dropped recently, coinciding with Microsoft's greater focus on security engineering processes. Firefox's code handling policies and lack of a strict security bug handling policy may have contributed to their relatively high number of vulnerabilities.

We conclude therefore that security in software, whether open source or not, depends on disciplined processes applied consistently throughout software design, implementation, testing and review. From our case studies, open source software is not as effective as closed source software in accomplishing this.