

**COMPARISON OF PROPRIETARY SOFTWARE AND OPEN
SOURCE SOFTWARE**

CASE STUDY: UGANDA DOT NET

BY

BUYONJO ANNIE MERCY

DCS/11418/61/DU

**A PROJECT REPORT SUBMITTED TO THE SCHOOL OF
COMPUTER STUDIES IN PARTIAL FULL FILLMENT**

FOR THE AWARD OF A DIPLOMA IN COMPUTER


SCIENCE OF KAMPALA INTERNATIONAL

UNIVERSITY

OCTOBER 2009

DECLARATION

I **Buyonjo Annie Mercy** do hereby declare that this research is my original work and that it has never been submitted to any academic institution for the award of a diploma or its equivalent

Signature 

Date 11th Sept 2009

APPROVAL

This is to certify that this project report has been done under my supervision and is now ready for submission to the board of examiners with my approval.

Signature  Date 11th 09-09

Supervisor: Mr. Chemutai Gilbert

ABSTRACT

Proprietary software are programs whose licenses give the user permission to run them but are not allowed to share, alter or even redistribute them while Open Source Software/Free Software (OSS/FS), also abbreviated as FOSS (Free Open Source Software) licenses give users the freedom to run the program for any purpose, study, modify, and redistribute copies of either the original or modified program without having to pay royalties to previous developers. Proprietary software has been the most commonly used software among the public over the years though with the growth of the Information and Communications Technologies (ICTs) industry, FOSS is a rapidly growing and commercially accepted alternative to proprietary software in the world. In Uganda, Proprietary software is the mostly used though FOSS is winning several users too, but most users do not purchase licenses for these software products and have little or no information about benefits or challenges of using either software. This research aims at correcting this problem by providing quantitative and qualitative results that users can use to compare software and thereby purchase appropriate programs for their organizations. Therefore the choice of OSS/FOSS and proprietary software is an issue whose importance can not be ignored. Our research reveals that OSS/FOSS and proprietary performances depend on the context and situation. When it comes to this parameter, it is hard for one to conclusively say that FOSS or proprietary software is better. The research, however, reveals that FOSS is more reliable and more secure than the proprietary software.

TABLE OF CONTENTS

DECLARATION.....	i
APPROVAL	ii
ABSTRACT	iii
TABLE OF CONTENTS	iv
 CHAPTER ONE.....	 1
INTRODUCTION.....	1
1.0General Introduction	1
1.1 Background	2
1.3 Background of FOSS.....	2
1.4Foss in Africa	3
1.5 Background to the Case Study	3
1.6 Problem Statement	4
1.8 Research Questions	5
1.9 Scope of the Research.....	5
1.10 Significance Justification of the Study	5
 CHAPTER TWO	 7
LITERATURE REVIEW	7
2.0 Introduction.....	7
2.1 FOSS and Proprietary Software	7
2.2 Open Standards	8
2.3 Reliability	9
2.4 Performance.....	11
2.5 Security	13
 CHAPTER THREE	 26
RESEARCH METHODOLOGY	26
3.0 Introduction.....	26
3.1 Research Design.....	26
3.2 Sources of Data.....	26
3.3 Data Collection Tools	26
3.4 Population of the Study	27
3.6 Data Analysis Tools	28
3.7 Data Presentation	28
3.8 Limitations of the Study.....	28
 CHAPTER FOUR	 30
DATA ANALYSIS AND PRESENTATION OF RESULTS	30
4.0 Introduction.....	30
4.1 Results from the Survey	30
4.2Reliability	31
4.3 Software.....	31
4.4 Performance.....	31

4.5 The Case Study.....	33
4.6 Results from Case Study Observation.....	34
4.7 Results from Interview Guide.....	35
 CHAPTER FIVE	37
DISCUSSION, CONCLUSION AND RECOMMENDATIONS	37
5.0 Introduction.....	37
5.2 Discussion	37
5.3 Conclusion.....	40
5.4 Recommendations	41
REFERENCES	44

CHAPTER ONE

INTRODUCTION

1.0 General Introduction

Proprietary software also known as “closed” software are programs whose licenses give users permission to run them but are not allowed to share alter or even redistribute them without permission. While Open Source Software/Free Software (*OSS/FS*), also abbreviated as FOSS (Free Open Source Software) or FLOSS (Free Libre Open Source Software), licenses give users the freedom to not only to run them but also to study, modify, and redistribute copies of either the original or the modified program without having to pay royalties to previous developers. Proprietary software has been the most commonly used software among the public over the years Microsoft Windows Vista, **XP**, Internet Information Server (IIS), Internet Explorer, Oracle, Opera, Sun Solaris, Mac OS, and Unix among others. While, Free or non-proprietary software, a rapidly growing and commercially accepted alternative to proprietary software today includes GNU/Linux, Apahe Web Server, Mozilla Firefox, and Proprietary software includes. The aim of this research is to analyze reliability, performance and security of both FOSS and proprietary software, (so one can consider using either software when looking for software), using quantitative and non-quantitative measures to justify why using either FOSS or proprietary software products is a more a reasonable approach over the other.

1.1 Background

Though proprietary software is a market stronghold, Open Source Software is increasingly cited as a viable alternative to commercial proprietary software, with

Potential significant value for money benefits for organizations. It is based principle of software products made available by the FOSS developer community includes commercial companies, academics and others) licensed for use with or a fee. FOSS licenses generally give the user the freedom to use, copy, distribute, ex change and improve the software. The commercial models that underpin distribution typically include support charges, but there can also be other cost benefits perceived to be associated with FOSS deployment. Costs may include training, migration of existing files and applications, and the effort required to integrate with other software. Apart from reductions in the cost of software licenses, benefits of OSS/FOSS include cost avoidance through reductions in replacement cycles of hardware, imp software reliability and security, software platform stability, the ability to tailor and the software, easier administration, and greater scalability of hardware platforms.

1.3 Background of FOSS

Free or non-proprietary software has existed since the invention of the first computers in the mid-1940s, and for many years it was the norm. In the 1970s, programme AT&T Bell labs developed the UNIX operating system using the C program language, which could be ported to any proprietary machine.

Later, AT&T began to increase the licensing fees for its UNIX operating system, le to an initiative by Richard Stallman at MIT's Artificial Intelligence Laboratory against proprietisation of software. Stallman's pioneering efforts with the GNU project paved way for the development of GNU/Linux .an open UNIX-like alternative _which released in 1994 after Linus Torvalds' contribution of an operating system kernel ft GNU project resulted in a complete system. Today, FOSS is a rapidly growing commercially accepted alternative to proprietary software across a range of systems and applications (VVheeler, 2007).

1.4 Foss in Africa

Recently, the Free Software and Open Source Foundation for Africa (FOSSFA) and a number of partner organizations convened a conference that was content rich and brought together a diverse range of people working in the areas of FOSS and Open Content and saw to the release of a Kiswahili spellchecker. In Uganda, a mailing list has been put in place for Linux Users (those working in the IT field in Uganda and are interested in and use Open Source Software). Plans are also underway to conduct regular meetings of Linux Users in Uganda and the monthly I- Network seminar series for a day dedicated to Open Source Software (OSS) where the general public, university students and people from government and industry are invited to participate in information sessions and demonstrations of FOSS.

1.5 Background to the Case Study

1.5.1 Uganda Dot Net

Most government ministries in Uganda and ISPs were running on GNU/Linux, but support personnel were being flown into the country to operate these installations from abroad, at great cost. This lack of capacity within the country was addressed by IICD (International Institute for Communication and Development) from Netherlands in a joint venture with Uganda Martyrs University and Linux Solutions Ltd, and resulted in the establishment of a Non-Governmental Organization (NGO) called the for Uganda Dot Net. UGANDA DOT NET, located in parliamentary Avenue, Kampala was established in 2008 as a software development organization, aimed at raising awareness of the benefits of FOSS, and providing essential training for system and network administrators in Uganda. Uganda Dot Net's immediate goal is to ensure the spread and taking on of OSS/FOSS in the Uganda and have since then been involved in FOSS advocacy and training. They have a regular event, called the Free Software Weekend where people are invited talk to them, and they even offer free training to these users. Uganda Dot Net has penetrated many ICT organizations which invite them for quite a number of talks. Most

of their PCs in their organization run FOSS products since they are major advocates of FOSS and trainers.

1.6 Problem Statement

Open Source Software and proprietary software use in the ICT industry is growing steadily side-by-side but there is a need to find out the challenges and benefits of each software such that one can objectively make a choice of the appropriate software to use for a particular purpose so as to effectively meet the objectives of the organization. This research seeks to solve this problem by analyzing reliability, performance and security of both OSS/FS and proprietary software that one can consider using *OSSIFS* or proprietary software when looking to purchase software to meet their organizational needs.

1.7 Objectives

General Objective The main objective of this research is to analyze the reliability, performance and security of both FOSS and proprietary software so one can consider using either FOSS or proprietary software.

Specific Objectives

1. To investigate the functionality of both FOSS and proprietary software.
2. To measure the quality of ICT services when using FOSS or proprietary software so that users can choose appropriate software given information about its benefits and challenges.
3. To compare the implementation and operational costs of each software such that users can choose the most cost effective software for their organization
4. To eliminate software piracy that arises from inability of the users to purchase expensive software licenses by providing the public with knowledge about FOSS.

1.8 Research Questions

This research will attempt to answer the following question in an organizational setting:

Which software platform: OSS/FLOSS and/or Proprietary Software offers a high level of performance, security and reliability?

To further guide our research, the following sub-questions will be used:

1. what are the challenges and benefits of using either OSS/FOSS proprietary software?
2. What software products (OSS/FOSS or proprietary) are more reliable, secure and have better performance features?

1.9 Scope of the Research

This research will attempt only to comparatively analyze software reliability, performance and security of the most commonly-used FOSS and proprietary software, to show that by certain measures, some FOSS or proprietary software is as good as or even better than its competition. The term “better” in the context of this research refers to the ability of the software to meet the users specific needs. However good software is, once it does not meet the user’s requirements, it is technically poor for that purpose. Of course, some FOSS software is technically poor, just as some proprietary software is technically poor.

Emphasis will be made on server software and operating systems and web browsers, especially some of the most visible FOSS projects such as GNU/Linux (or Linux) operating system (OS), the Apache web server, Mozilla Firefox as compared to proprietary products such as Windows operating systems, Internet Information server (IIS) and Internet Explorer.

1.10 Significance Justification of the Study

This research emphasizes quantitative measures (such as experiments and observations) as well non-quantitative measures (opinions) to justify why using either FOSS or proprietary software products is in many circumstances a reasonable or even superior

approach over the other. It is hoped that the findings of the study will enable one to objectively consider using *OSSIFS* or proprietary software when looking for software basing on reliability, performance and security as well as provide more information on the performance, reliability and security of both FOSS and proprietary software. The beneficiaries of this study are UGANDA DOT NET who use the software for studying, training and other day to day activities. UGANDA DOT NET do provide training in FOSS and so require information on how this software can be implemented and used more effectively to achieve their individual goals.

CHAPTER TWO

LITERATURE REVIEW

2.0 Introduction

This chapter discusses the review of previous works by various authors and scholars prior to this research. Sections 2.1 and 2.2, clarify the terminologies used in this research, while sections 2.3, 2.4 and 2.5, discuss previous comparative studies on reliability, performance, and security of both FLOSS and Proprietary software respectively.

2.1 FOSS and Proprietary Software

Accounting to Damien Challet he refers to Software as carries the instructions that tell a computer how to operate. The human authored and human readable form of those instructions is called *source code*. Before the computer can actually execute the instructions, the source code must be translated into a machine readable (binary) format, called the *object code*. All distributed software includes the object code, but FLOSS makes the source code available as well. Proprietary software owners license their copyrighted object code to a user, which allows the user to run the program. FLOSS programs, however, license both the object and the source code, permitting the user to run, modify and possibly redistribute the programs. With access to the source code people have the freedom to run the program for any purpose, redistribute, probe, adapt, learn from, customize the software to suit their needs, and release improvements to the public for the good of the community. Open source software (OSS) refers to technical advantages of such software (for instance better reliability and security), while Free Software (FS) refers to freedom from ethical Other alternative terms for OSS/FS include “libre or “livre” software (where libre or livre means freedom),

free-libre / open-source software (FLOSS), free / open source software (FOSS or F/OSS). FLOSS will be used in this research because it's easier to pronounce. All FOSS developers claim copyright, but then use licenses innovatively to give users a variety of freedoms. Dominant examples of these licenses are copylefted and noncopylefted (Lessig, 2003). Under the copylefted license, subsequent FLOSS developers must adopt the same license which ensures that their modifications to the code will remain open. With the non-copylefted license, developers can choose any license to cover their subsequent modifications, even a proprietary license.

2.2 Open Standards

Paul Murphy emphasized that Open standards in Information and Communications Technologies (ICT5) allow freedom to access as well as to contribute to the development of the standard by any interested party, which is not possible under a proprietary standard. Standards are *de facto* and *de jure*, and both arise out of complex dynamics influenced by economic, political and social forces. Open standards in ICTs are critical to allow new entrants to participate, innovate on standards implementation, and compete. With a proprietary standard, the owner can prevent competitors and entrants from capturing market share through their legally enforceable IPR. In a developing economy like Uganda, proprietary ICT standards are typically held by foreign enterprises which effectively relegate domestic engagement to the level of franchisee.

It cannot be over-emphasized that the Internet and World Wide Web, which are having so much impact on the world today, would not exist without FOSS and the inter-i development and adoption of the essential open standards. Today, nearly two-thirds of all web servers use FOSS. A FOSS program must be released under some IL giving its users a certain set of rights; the most popular FOSS license is

the GNU General Public License (GPL). All software released under the GPL is FOSS, but not all FOSS software uses the GPL (Wheeler, 2007). In Uganda, Inveneo, a San Francisco non-profit, together with Action Aid, an international agency whose aim is to fight poverty worldwide, installed their first rugged Linux desktop systems in western Uganda. The systems run Linux and KDE desktops, and also include the OpenOffice.org productivity suite (Kendrick, 2005).

“The system is up and running since this June, 2005 where 5 units have been installed, four of which are in villages with no access to power. The system provides Internet access and phone capabilities to the users. Phone calls among the connected villages are free of charge, with the ability to place and receive calls to the Ugandan phone network. The systems are linked using 80211 WiFi links.” (Summer, 2005).

2.3 Reliability

Reliability as popularly measured by how long software can resist crashing or freeze-ups from the time it is installed. For instance paper ‘Fuzz Revisited’ measured reliability by feeding programs random characters and determining which ones resisted crashing and freeze-ups. defined by Justin E.Forrester and Barton P.Miller FOSS had higher reliability by this measure. A later paper published in 2000, “An Empirical Study of the Robustness of Windows NT Applications Using Random Testing”, found that with Windows NT GUI applications, they could crash 21% of the applications they tested, hang an additional 24% of the applications, and could crash or hang *all* the tested applications when subjecting them to random WIn32 messages. The fuzz paper’s authors also found that proprietary software vendors generally didn’t fix the problems identified in an earlier version of their paper (from 1990). There was a slight decrease in failure rates between their 1990 and 1995 paper, but many of the flaws they found (and reported) in the proprietary Unix programs were still not fixed 5 years later. In contrast, Scott Maxwell led an effort to remove every flaw identified in the

OSS/FS software in the 1995 fuzz paper, and eventually fixed every flaw. Thus, the OSS/FS community's response shows why, at least in part, OSS/FS programs have such an edge in reliability; if problems are found, they're often fixed. IBM put Linux to a 10-month test and ran Caldera Systems Open Linux, Red Hat Linux, and Windows NT Server 4.0 with Service Pack 3 on duplicate 100MHz Pentium systems with 64MB of memory. The results: the NT server crashed an average of once every six weeks. Each failure took roughly 30 minutes to fix. That is really bad considering that neither Linux server ever went down. (Vaughan-Nichols, et al, 1999). A 3-month Swiss evaluation, on the difference between Netscape (proprietary) and Apache (FOSS) shows the results of Syscontrol AG's analysis of website uptime (announced February 7, 2000) They measured over 100 popular Swiss web sites over a three-month period, checking from 4 different locations every 5 minutes. Below is their set of published data on "average down-time (in hours in that month) for each type of server", plus a 3-month average that I have computed:

German import company Heinz TrOber compared Linux-based desktops to Windows and found that Windows had a 15% daily failure rate, while Linux has 0%. ...ter Stoverock, the data processing manager at German import company Heinz r, reported that they had decided to run its ERP software on Linux-based systems, of Windows, because Windows was much less reliable. Stoverock stated that in Windows, "Out of 65 desktops, around 10 desktops crashed daily. Employees around 30 minutes, that's five times 30 minutes per week." Note that this is a daily failure rate, and the actual impacts were almost certainly more severe than a loss of 2 minutes of lost time per reboot. imien Challet and Yann Le Du of the University of Oxford have written a paper titled Closed source versus open source in a model of software bug dynamics, where they develop a model of software bug dynamics in which users, programmers can interact through a given program. Then they analyzed the model, and found that all other things being equal (such as

number of users, programmers, and quality of programmers), “debugging in open source projects is always faster than in closed source projects.” (Damien, et al, 2005).

However, one problem with reliability measures is that it takes a long time to gather data on reliability in real-life circumstances. Thus, there’s more data comparing older Windows editions to older GNU/Linux editions. The key is that these comparisons are fair, because they compare contemporaneous products. (Wheeler, 2007).

2.4 Performance

Birabwa K (2006) defines open software as a measure of performance of software, it is best to set up the benchmark yourself in a given environment, as performance is very sensitive to the assumptions and the environment because it is very likely that you may use biased measures from other authors. However below are different findings.

On comparing GNU/Linux and Microsoft Windows performance on equivalent hardware, TPC-C database (2002) measure found that a Linux based system on HP ProLiant DL580 with 32 Intel Xeon 900MHz CPUs running Oracle 9i R2 Enterprise edition ran faster running on a stock Red Hat Linux Advanced Server than on Microsoft Windows 2000 Advanced Server. According to the report, HP did not modify the Linux kernel to get these results as illustrated in these two performance tests: Their “real-world” test measured how quickly large quantities of email could be sent using their email delivery server (MailEngine). Up to 100 simultaneous sends there was no difference, but as the number increased the systems began showing significant differences in their hourly email delivery speed. By 500 simultaneous sends GNU/Linux was clearly faster than all except FreeBSD-tuned, and GNU/Linux remained at the top. FreeBSD-tuned had similar performance to GNU/Linux when running 1000 or less simultaneous sends, but

FreeBSD-tuned peaked around 1000-1500 simultaneous connections with a steady decline not suffered by GNU/Linux, and FreeBSD-tuned had trouble going beyond 3000 simultaneous connections. By 1500 simultaneous sends, GNU/Linux was sending 1.3 million emails/hour, while Solaris managed approximately 1 million, and Windows 2000 and FreeBSD-untuned were around 0.9 million. Their “disk I/O test” created, wrote, and read back 10,000 identically-sized files in one directory, varying the size of the file instances. Here Solaris was the slowest, with FreeBSD-untuned the second-slowest. FreeBSD-tuned, Windows 2000, and GNU/Linux had similar speeds at the smaller file. When totaling these times across file sizes, the results were GNU/Linux: 542 seconds, Windows 2000: 613 seconds, FreeBSD-tuned: 630 seconds, FreeBSD-untuned: 2398 seconds, and Solaris: 3990 seconds. One organization that tries to develop unbiased benchmarks is the SPEC Consortium, which develops and maintains a whole series of benchmarks. We can compare Microsoft Windows versus GNU/Linux by comparing SPECweb99 results (which measure web server performance) on identical hardware if both have undergone the same amount of performance optimization effort. Using all results available by July 13, 2001, there were three hardware configurations, all from Dell, which ran both GNU/Linux (using the TUX web server/accelerator) and Windows (using uS) on exactly the same underlying hardware. The SPECweb99 results as of July 13, 2001 (larger is better), noting configuration differences are as follows: The second entry (the PowerEdge 6400/700)

certainly suggests that GNU/Linux plus TUX really is much better. the IIS system had two more disk drives available to it (which should increase performance), but the TUX system had over twice the us system’s performance. The IIS systems had at least one drive that revolved more quickly than the TUX systems. Note that TUX is intended to be used as a “web accelerator” for many circumstances, where it rapidly handles simple requests and then passes more complex queries to another server (usually Apache).

Basing on Windows and GNU/Linux pipes (an input/output mechanism), process thread creation, a study examined the performance of pipes, a common low-level mechanism for communicating between program processes and it was found that the pipes in Red Hat 7.1 (with Linux kernel version 2.4.2) had a peak I/O rate of around 1 MB/sec, with a steady state at near 100 MB/sec for very large block sizes. In contrast, Windows 2000 peaked at 500 MB/sec, with a large block steady state of 80 MB/sec. Windows XP Professional peak I/O rate was only 120 MB/sec, with a steady state of 80 MB/sec. Comparing the performance of threads; Linux managed to create over 10,000 threads/second, while Win2K did not quite manage 5,000 threads/second and Win XP only created 6,000 threads/second. In process creation, Linux managed 330 processes/second, while Win2K and WinXP created less than 200 processes/second and 160 processes/second respectively (Wheeler, 2007). A report that compares their research prototype to Windows, Linux, and FreeBSD exposes performance figures that compare these operating systems directly to each other. What's noteworthy about it is that Microsoft compared Singularity to FreeBSD and Linux as well as Windows/XP and almost every result shows Windows losing to the two Unix variants." (Murphy, 2005).

2.5 Security

quantitatively measuring security is very difficult. Security is it is often measured by a survey of opinions, from people who are informed and have used the software because it is hard to measure. Though opinions can be wrong it is very hard to ignore the opinions collected from a large sample space of users who are in the know. This research will concentrate on comparing FOSS to Windows systems, since other proprietary systems are increasingly including FOSS components (making comparisons more difficult).

At one time the security of FOSS systems was widely debated. Clearly FOSS systems are not magically invincible from security flaws. But for most of those who study the question, the issue of whether or not FOSS improves or reduces security appears to be an increasingly settled issue. Below are some quantitative studies that compare this

software.

According to a combination of studies from the Honeynet Project (2004), AOL, and others that compared the unpatched Linux systems to unpatched Windows systems, the average Linux system lasts three months before being compromised, (a significant increase from the 72 hours life span of a Linux system in 2001). Unpatched Windows systems continue to be compromised far more quickly, sometimes within minutes. This data on Windows compromise is consistent with other studies. Avantgarde found that Windows did not last long, and one unpatched Windows XP system (pre-SP2) only lasted 4 minutes on the Internet before it was compromised and in general did not last long. Note, however, that users who install Windows Service Pack 2 have much less risk than previous versions of Windows. Symantec's Internet Security Threat Report (January 1-June 30, 2004), The Internet Storm Center's Survival Time History claims that by December 2004 a Windows survival time of 18 minutes. The Bugtraq vulnerability database that examines security was used to compare the vulnerability of OSes running FOSS and proprietary software in 1999-2000. One approach to examining security is to use a vulnerability database. Below is an analysis of one database from the Bugtraq Vulnerability Database Statistics page as of September 17, 2000, listing the total number of vulnerabilities for some leading OSes:

Windows NT12bdO

Some vulnerabilities are more important than others (some may provide little if exploited or only be vulnerable in unlikely circumstances), and some vulnerabilities are being actively exploited (while others have already been fixed before exploitation). Comparing the FOSS and proprietary response to security problems; Red Hat (an FOSS vendor) responded more rapidly than Microsoft or Sun to advisories. Sun had fewer advisories to respond to yet took the longest to respond. Security Portal compiled a list of "the time it takes for vendors to respond to vulnerabilities", and concluded that Red Hat had the best score, with 348 recess days on 31 advisories, for an average of 11.23 days from bug to patch. Microsoft had 982 recess days on 61 advisories, averaging 16.10 days from bug to patch. Sun proved itself to be very slow, although having only 8 advisories it

accumulated 716 recess days (three months) to fix each bug on average. Their table of data for 1999 is as shown:

Vendor	Total Days, Hacker Recess	Total Advisories	Recess Days Advisory
Red Hat	348	31	11.23
Microsoft	982	61	16.10
Sun	716	8	89.50

It is worth noting that the Open BSD OS (FOSS) had fewer reported vulnerabilities than all of these. Clearly, having a proprietary OS doesn't mean you're more secure. Microsoft had the largest number of security advisories, by far, using either counting method. Eweek Labs' article "Open Source Quicker at Fixing Flaws" (September 30, 2002) listed specific examples of more rapid responses. This article can be paraphrased as follows: In June 2002, a serious flaw was found in the Apache Web server; the Apache Software Foundation made a patch available two days after the Web server hole was announced. In September 2002, a flaw was announced in OpenSSL and a patch was available the same day. In contrast, a serious flaw was found in Windows XP that made it possible to delete files on a system using a URL; Microsoft fixed this problem in Windows XP Service Pack 1 without notifying users of the problem. A more direct comparison can be seen in how Microsoft and the KDE Project responded to an SSL (Secure Sockets Layer) vulnerability that made the Internet Explorer and Konqueror browsers, respectively, potential tools for stealing data such as credit card information. The day the SSL vulnerability was announced, KDE provided a patch.

In an August 18, 2004 interview, Symantec's chief technology officer Robert Clyde argued that proprietary vendors were more reliable for fixing problems within a fixed timescale, and that he didn't know of a single vendor who would sit on a vulnerability. Yet the day before (August 17), and eWeek article revealed that Oracle waited 8 months

to fix a vulnerability. And Microsoft waited 9 months to fix a critical IE vulnerability (and only fixed it after it was being actively exploited in 2004). A study on software immunity from outside attacks by Evans Data Corp.'s (2002), over 400 GNU/Linux developers found that even though computer attacks have almost doubled annually since 1988 (according to CERT), 78% of the respondents to the GNU/Linux developers survey have never experienced an unwanted intrusion and 94% have operated virus-free. Clearly, the survey shows that GNU/Linux "doesn't get broken into very often and is even less frequently targeted by viruses," according to Jeff Child Evans Data Corp.'s Linux Analyst who tested Linux systems immunity from attacks from outsiders notes that it's much harder to hack a knowledgeable owner's system (and most Linux developers have hands-on, technical knowledge) and that because there are fewer desktop GNU/Linux systems there are fewer viruses being created to attack GNU/Linux. The developers being surveyed attributed the low incidence of attacks to the Open Source Software (OSS) environment; "more than 84% of Linux developers believe that Linux is inherently more secure than software not created in an OSS environment," and they ranked "Linux's security roughly comparable in security to Solaris and AIX ... and above any of the Windows platforms by a significant margin." Eweek's report that compared the security of Apache to Microsoft's IIS, examined that Apache's last serious security problem (one where remote attackers could run arbitrary code on the server) was announced in January 1997. A group of less serious problems (including a buffer overflow in the server's logresolve utility) was announced and fixed in January 1998 with Apache 1.2.5. In the three and a half years since then, Apache's only remote security problems have been of denial-of-service and information leakage problems (where attackers can see files or directory listings they shouldn't). eWeek's April 10, 2002 article noted that ten more IIS flaws were found in IIS Server 4.0, 5.0, and 5.1, some of which would allow attackers to crash the IIS service or allow the attacker to run whatever code he chooses.

Apache wisely follows the good security practice of “least privilege.” While IIS is designed so that anyone who takes over IIS can take over the whole system, performing actions such as reading, modifying, or erasing any file on the system. In contrast, Apache is installed with very few privileges by default, so even taking over Apache gives attackers relatively few privileges. For example, cracking Apache does not give attackers the right to modify or erase most files.

The article claims there are four reasons for Apache’s strong security, and three of these reasons are simply good security practices. Apache installs very few server extensions by default (a “minimalist” approach), all server components run as a non-privileged user (supporting “least privilege” as noted above), and all configuration settings are centralized (making it easy for administrators to know what’s going on). However, the article also claims that one of the main reasons Apache is more secure than IIS is that it’s “source code for core server files is well-scrutinized,” a task that is made much easier by being FOSS, and it could be argued that OSS/FS encourages the other good security practices.

It was attacked 1,400 times more frequently than Apache in 2001, and Windows was attacked more than all versions of UNIX. SecurityFocus co-founder and CEO Arthur Wong reported an analysis of the various vulnerabilities and attacks (based on SecurityFocus’s data) in the February 2002 article. It was attacked 17 million times, but Apache was attacked only 12,000 times. In 2001, Windows systems were attacked 31 million times, while Unix systems were attacked 22 million times.

Some security vulnerabilities are more important than others, for a variety of reasons. Thus, some analysis centers try to determine what’s “most important,” and their results suggest that OSS/FS just doesn’t have as many vulnerabilities.

The CERT Coordination Center (CERT/CC) is federally funded to study security vulnerabilities and perform related activities such as publishing security alerts. Four of the six most important security vulnerabilities were specific to Microsoft: W32/Nimda, W32/Sircam, cache corruption on Microsoft DNS servers, and “Code

Red” related activities. Only one of the six items primarily affected non-Microsoft products (a buffer overflow in telnetd); while this vulnerability is important, it’s worth noting that many open source systems (such as Red Hat 7.1) normally do not enable this service (telnet) in the first place and thus are less likely to be vulnerable. Computer viruses are a serious security problem in software. Virus infection has been a major cost to users of Microsoft Windows. The LoveLetter virus alone is estimated to have cost \$960 million in direct costs and \$7.7 billion in lost productivity, and the anti-virus software industry sales total nearly \$1 billion annually. Dr Nic Peeling and Dr Julian Satchell’s *Analysis of the Impact of Open Source Software* includes an analysis of the various data sources for virus counts, noting the disproportionate vulnerability of Windows systems.

They found out that numbers differ in detail, but all sources agree that computer viruses are overwhelmingly more prevalent on Windows than any other system. There are about 60,000 viruses known for Windows, 40 or so for the Macintosh, about 5 for commercial Unix versions, and perhaps 40 for Linux. Most of the Windows viruses are not important, but many hundreds have caused widespread damage. Two or three of the Macintosh viruses were widespread enough to be of importance. None of the Unix or Linux viruses became widespread .most were confined to the laboratory.

In contrast, while it’s possible to write a virus for FOSS OSes, their design makes it more difficult for viruses to spread... showing that Microsoft’s design decisions were not inevitable. It appears that FOSS developers tend to select design choices that limit the damage of viruses, probably in part because their code is subject to public inspection and comment (and ridicule, if deserving of it). For example, FOSS programs generally do not support attacker-controlled start-up macros, nor do they usually support easy execution of mail attachments from attackers. Also, leading OSS/FS OSes (such as GNU/Linux and the *BSDs) have always had write protection on system directories, making it more difficult for certain attacks to spread. OSS/FS systems are *not* immune to malicious code, but they are certainly more resistant.

According to a June 2004 study by network management firm Sandvine, *Over 90% of all spam comes from computers contaminated with Trojan horse infections. Trojans and worms with backdoor components turn infected PCs into drones in vast networks of 'unpromised' zombie PCs. Most PCs affected were running Windows programs. Sandvine identified subscribers bypassing their home mail servers and contacting many mail servers within a short period of time over sustained periods .i.e., spammers. It also looked at SMTP error messages returned to clarify the total volume of spam. They then compared this with the messages passing through the service provider's mail system. National Cyber Security Alliance's study of May 2003 reported that 91% of Broadband users have spyware on their home computers running proprietary operating systems. In contrast, there's no evidence of that this is an issue for OSS/FS systems. America Online, Inc. conducted a study for the National Cyber Security Alliance. Its results, "Fast and Present Danger: In-Home Study on Broadband Security among American Consumers" (May 2003). They found that "91% of Broadband Users Have Spyware Lurking on Home Computers". Their study method did not appear to permit collection of data from OSS/FS systems, and spyware systems are essentially nonexistent on OSS/FS systems anyway. The SecurityTracker Statistics paper (March, 2002) analyzed vulnerabilities from April 2001 through March 2002 and identified 1595 vulnerability reports, covering 1175 products from 700 vendors. Their analysis found that Microsoft had 187, or 11.7% of all vulnerabilities), and more than four times the next vendor. The next largest were Sun (42, 2.6% of the total), HP (40, 2.5%), and IBM (40, 2.5%). Solely OSS/FS vendors did much better: the Apache Software Foundation had 13 (0.8% of the total), and Red Hat had 10 (0.6% of the total). In late June 2004, Microsoft had failed to patch a critical vulnerability for 9 months, and IE was being actively exploited in horrendous ways. Customers then rushed to download Mozilla and Mozilla Firebird, popular OSS/FS alternatives, to replace IE. The U.S. CERT warned that the Microsoft browser (IE) cannot protect against vulnerabilities,*

The "quantity" shows the number of vulnerabilities, but doesn't account for their criticality. Thus, he also computes a "relative danger" by simply "adding up the

criticality levels for each vulnerability (not critical=1, extremely critical=5)". As of that date:

- "Internet Explorer has had 43 reported vulnerabilities. 7 were marked as moderately critical, 11 were marked as highly critical, and 6 were marked as extremely critical. There are still 25 unfixed vulnerabilities, including 6 that were marked as moderately critical, 1 that was marked as highly critical, and 1 that was marked as extremely critical."
- "Mozilla Firefox has had 21 reported vulnerabilities. 8 were marked as moderately critical, 4 were marked as highly critical, and 0 were marked as extremely critical. There are still 4 unfixed vulnerabilities, including 1 that was marked as moderately critical."
- "Opera has had 23 reported vulnerabilities. 14 were marked as moderately critical, 0 were marked as highly critical, and 0 were marked as extremely critical. All reported vulnerabilities have since been fixed."

Brian Krebs "Security Fix" column compiled statistics on vulnerability response times, including those for Microsoft Internet Explorer (IE) and Mozilla Firefox. He found that for "a total 284 days in 2006 (or more than nine months out of the year), exploit code for known, unpatched critical flaws in pre-IE7 versions of the browser was publicly available on the Internet. Likewise, there were at least 98 days last year in which no software fixes from Microsoft were available to fix IE flaws that criminals were actively using to steal personal and financial data from users.

In TechWeb.com (February 9, 2005), Gregg Keizer's article "Spyware Barely Touches Firefox" describes some research work from the University of Washington. Henry Levy stated that his research showed that users "will have a safer experience [surfing] with Firefox." Researchers Henry Levy and Steven

Gribble crawled 45,000 websites, cataloguing their executable files, and then exposed unpatched Internet Explorer (IE) and Firefox browsers to them. "If you browse, you're eventually going to get hit with a spyware attack." (Levy) Perhaps choosing the program with the better record would help. Microsoft took 134 days on average to release patches for security problems in 2004- 2005; Mozilla averaged 37 days. Brian Krebs' "A Time to Patch II: Mozilla" compared patch times of Mozilla with Microsoft. Even with an outlier included, Mozilla did much better on average than Microsoft. Mozilla took an average of about 37 days to issue patches for critical security problems in its products over a 3-year period. In general it did much better; one-third of its critical security updates were within less than 10 days of being notified. (The longest time was for a bug that perhaps should not have been marked as "critical"; Microsoft had exactly the same bug but marked it only as moderate.) In a similar study of Microsoft's vulnerability report response times, he notes that "In 2003, Microsoft took an average of three months to issue patches for problems reported to them. In 2004, that time frame shot up to 134.5 days, a number that remained virtually unchanged in 2005." Christian Payne's *Information Systems Journal*, Vol.12, Issue 1, February 2002, includes the peer-reviewed paper "On the security of open source software" by Christian Payne of Murdoch University (Perth, Australia). In it, Payne first summarizes the various arguments made for and against open source software. He discusses some of the arguments that FOSS is more secure, in particular, claims that the process of peer review improves security, FOSS flexibility and freedom is a significant aid (e.g., organizations are free to audit FOSS, modify it to meet their security needs, and rapidly patch OSS/FS without having to wait for a vendor), and that OSS/FS projects tend to respond more quickly with security fixes. He also discusses some of the arguments made against OSS/FS, such as claims that that vulnerabilities are harder for attackers to find in proprietary programs (since the source code is not available), and that there are flaws in the peer review argument (e.g., it may be available but not necessarily reviewed). In short, there are different effects, and

it's easy to have opinions about the strengths of those different effects. Without measurement, it's hard to know what effects are stronger. But Payne goes beyond a mere summary of arguments, and actually works to try to gather quantitative data to measure the effect of these alternative approaches. Payne devised a scoring system for measuring security features, measuring reported security vulnerabilities, and then rolling those two factors into a final score. He then applied this to two *OSS/FS* systems (Debian and OpenBSD) and one proprietary system (Solaris, which at the time was proprietary); all are Unix-based operating systems. Table 2.7: Security of OSS/FS (Wheeler, 2007) OpenBSD had the most security features (features that support confidentiality, integrity, availability, or audit), with Debian second and Solaris third. OpenBSD also had the highest score for those features. In terms of vulnerabilities, OpenBSD had the fewest reported vulnerabilities, and those vulnerabilities "were also relatively minor[,] only rating an average of 4.19 out of 10". Solaris, the proprietary system, had the largest number of vulnerabilities. The final rolled-up score is quite intriguing: of the three systems, the proprietary system had the worst security by this rolled-up measure. The author correctly notes that these are only a few systems, using information taken at only one point in time, so these results are "far from being final". And the author certainly does not take the view that any OSS/ES program is automatically more secure than any proprietary alternative. Hiding the source code certainly did not reduce the

A BZ Research survey of 6,344 software development managers in April 2005 asked about the security of different popular enterprise operating environments; OSS/FS did very well. Below are some of the results; the margin of error for the survey is 2.5 percentage points. number of reported vulnerabilities, contrary to some proprietary vendors' claims; the proprietary system had the most vulnerabilities reported about it.

Among server operating systems, there was uniform agreement that both Sun Solaris and Linux were much more secure than Microsoft's Windows Server against operating system related attacks. When comparing Sun Solaris against Linux by this measure, There was no consensus as to whether Sun Solaris or

Linux were better against operating system level attacks; more people ranked Linux as “secure or very secure” compared to Sun Solaris, yet more people also ranked Linux as “very insecure or insecure” than Sun Solaris.

Table 2.8: Comparing security of Windows, Linux and Solaris (Wheeler, 2007)

Windows Server also did poorly against application-related “hacks and exploits”:

Table 2.9: Comparing Windows to Linux (Wheeler, 2007)

number of reported vulnerabilities, contrary to some proprietary vendors’ claims; the proprietary system had the most vulnerabilities reported about it. A BZ Research survey of 6,344 software development managers in April 2005 asked about the security of different popular enterprise operating environments; *OSSIFS* did very well. Below are some of the results; the margin of error for the survey is 2.5 percentage points. Among server operating systems, there was uniform agreement that both Sun Solaris and Linux were much more secure than Microsoft’s Windows Server against operating system related attacks. When comparing Sun Solaris against Linux by this measure, There was no consensus as to whether Sun Solaris or Linux were better against operating system level attacks; more people ranked Linux as “secure or very secure” compared to Sun Solaris, yet more people also ranked Linux as “very insecure or insecure” than Sun Solaris.

Table 2.8: Comparing security of Windows, Linux and Solaris (Wheeler, 2007)

Windows Server also did poorly against application-related “hacks and exploits”:

Table 2.9: Comparing Windows to Linux (Wheeler, 2007)

	Ms Windows Server	Linux	Sun Solaris
Very Insecure or Insecure	58%	6%	13%
Secure or very secure	30%	74%	66%

OSS/FS was also far ahead of proprietary programs in 4 of the 8 categories considered: desktop/client operating systems (44% to 17%), Web servers (43% to 14%), server operating systems (38% to 22%), and components and libraries (34% to 18%). Results were essentially equal in three categories: desktop/client applications and application servers. Only in one area was proprietary software considered more secure than OSS/FS, database servers (34% to 21%). Security is notoriously hard to measure, and many reports that attempt to do so end up with interesting information that's hard to interpret or use. And some reports come from sources whose reliability is widely questioned. On November 2, 2004, mi2g reported on successful digital breaches against permanently connected computers worldwide. They concluded that BSDs (which are usually FOSS) and Apple's computers had the fewest security breaches; on the surface, that sounds positive for FOSS. They also reported that GNU/Linux systems had the most breaches, followed by Windows. That result sounds mixed, but digging deeper it turns out that this ranking is artificial, based on artificial definitions. Their default definition for a security breach only included manual attacks and ignored malware (viruses, worms, and Trojans). After all, why bother with a manual attack when completely automated attacks against broad collections of computers will do more? When they include malware in their calculations for all system breaches, "including the impact of MyDoom, NetSky, SoBig, Klez and Sasser, Windows has become the most breached computing environment in the world accounting for most of the productivity losses associated with malware .virus, worm and trojan .proliferation." Even without malware, in governments "the most breached Operating System for online systems has now become Windows (57.74%) followed by Linux (31.76%) and then BSD and Mac OS X together (1.74%)" (a reversal of their previous rankings). One of the most dangerous security problems with proprietary software is that if intentionally malicious code is snuck into it, such code is extremely difficult to find. Few proprietary vendors have other developers examine *all* code in great detail .their testing processes are designed to catch mistakes (not malice) and often don't look at the code at all. In contrast, malicious code can be found by anyone when the source code is publicly available, and with FOSS, there are incentives for arbitrary people to review it has to add new features or perform a security review of a product they intend to

use). Thus, someone inserting malicious code to an FOSS project runs a far greater risk of detection. Here are two examples, one confirmed, one not confirmed: Some time between 1992 and 1994, Borland inserted an intentional “back door” into their database server, “InterBase”, as a secret username and fixed password. This back door allowed any local or remote user to manipulate any database object and install arbitrary programs, and in some cases could lead to controlling the machine as “root”. This vulnerability stayed in the product for at least 6 years .no one else could review the product, and Borland had no incentive to remove the vulnerability. Then Borland released its source code on July 2000 as an OSS/FS project. The “Firebird” project began working with the source code, and uncovered this serious security problem with InterBase in December 2000 (only 5 months after release). By January 2001 the CERT announced the existence of this back door as CERT advisory CA-2001-01. Once this problem was found by open source developers reviewing the code, it was patched quickly. Note that this threat is unfortunately a credible threat to proprietary software, because very few of its users can review the code. This is far less dangerous to FOSS software, due to the worldwide review that’s possible (including the ability to see the changes made in each version). (Wheeler, 2007) This chapter has looked at the review of previous works by various authors prior to this research. We have clarified the terminologies used in this research and discussed previous works on comparative studies on reliability, performance, and security of both FOSS and Proprietary software respectively.

CHAPTER THREE

RESEARCH METHODOLOGY

3.0 Introduction

This chapter presents a background against which data was gathered and analyzed and reasons why those methods were chosen. The chapter provides a description of research design, sources of data, data collection methods, as well as data process and analysis.

3.1 Research Design

The study adopted a descriptive research design that involved both quantitative and non-quantitative primary/secondary data on the challenges of using FLOSS or Proprietary software in Uganda. Basing on this data, comparisons are then made between the software. This is consistent with the research objectives and questions in chapter one.

3.2 Sources of Data

The data sources were basically primary and secondary. Secondary sources included text books, journals, magazines, company records, seminar presentations, articles from newspapers and internet. Primary sources included tests, experiments, interviews and observations on ground.

3.3 Data Collection Tools

This research uses five methods for data collection that included the following Interview method which involved face-to-face interaction with the respondents and asking them questions in line with the research question. This method was chosen because it was the most effective way for the researcher get a first-hand

account of the information on the study especially on the reliability, performance and security of software.

Document review involved intensive researching and reading of different authors of literature about the variables of the study. This method was used as an additional source of information since the researcher could not get sufficient information on the functionality of the software from the respondents especially on comparing a wide range of software products. Observation and experiments involved the use of the eyes to see the challenges and benefits in the current system while also performing experimental tests the different software products and their alternatives. This method was used to get a clear objective view of the actual facts on ground especially at the case study.

3.4 Population of the Study

The following are categories of persons that participated in the above collection of data that involved key informants or personalities and users who interact very closely with the system and the organization: two ICT directors, two heads of departments, two systems administrators, one IT manager, eight lab technicians, four students and four other staff at the organizations as well as four people from the public at these institutions. The size of the sample was 10- 15 person in each institution.

3.5 Data Processing and Analysis

This was done by making references to the available literature in order to compare and contrast different opinions as presented by different authors. These opinions and quantitative data were then analyzed to arrive at the conclusions of the study as is reflected later in the findings.

3.6 Data Analysis Tools

Statistical Package for Social Scientists (SPSS), a statistical data analysis software tool; was used to process the quantitative data. Non-quantitative data was based on opinions expressed by the respondents, observation and experiments, was analyzed by comparison to achieve the conclusive results of the research.

Other tools used include: Hardware such as Laptops, Desktops, Server-ware, Network Interface Card as well as Software like Mozilla Firefox, Internet Explorer, Apache Web Server, Windows 2000, NT, 2003, XP and Vista, GNU-Linux products like Red Hat, Fedora, SuSe, and XandrOS.

3.7 Data Presentation

The findings and the information in this study were presented by use of quantitative and non-quantitative information through narration and providing the statistical results from the experimental tests. This research studies the works of other authors, discusses their opinions compares their analysis to the findings of the research through the experiments, observations and interviews performed by the researcher.

3.8 Limitations of the Study

Some respondents were not willing to disclose information especially about their core servers for security reasons and company policy. This was a major problem in the findings, as the researcher could not get sufficient information which could have gone a long way to shed more light on the study. The research was also limited by lack of equipment to use for experiments for instance acquiring a machine to crash so as to acquire actual facts was not easy. The researcher could not also get source code to experiment as most software available was not licensed. Also, the researcher had limited finances especially in purchase of stationary, printing and transport costs. This was

solved by proper budgeting while putting all the above mentioned considerations in mind. The time constraint was another limitation since the researcher had to study, teach and give tests, while compiling this report. But this was solved by proper following of the project time line. In this chapter, we have looked at the background against which data was gathered and analyzed as well as the reasons why those methods were used. We also provided a description of the research design, sources of data, data collection methods, data processing and analysis.

CHAPTER FOUR

DATA ANALYSIS AND PRESENTATION OF RESULTS

4.0 Introduction

This chapter analyzes interviews, observations and experiments carried out on the software performance, reliability and security. The chapter will also describe the results that were derived from the experiments and surveys. It also presents the information extracted from data collected during the study and the analysis derived from the findings. To make a comparative analysis of Open source software and proprietary software, an experimental survey rating of both sets of software was used where people in the industry were asked to give their opinion of each software, while sighting their reasons or each rating. The people interviewed were chosen from a sample space of 15 persons per institution. These people have spent 2 years or more in the ICT industry, working in key positions relevant to the study question such as system administrators, network administrators, instructors and lecturers, ICT directors plus 15 others from the public and student community to get a view of how both software would perform on average among those who are less informed about ICT. Test experiment was done by the researcher, on Mozilla Firebox and Internet Explorer; and OS Windows Vista and Windows 2003, to compare reliability, performance and security. These experiments were carried out to allow for the objective comparative analysis of how both software would perform in organizations and then finally compare the case study environment to the Industry standards.

4.1 Results from the Survey

These results are based on users' opinions which can be quite wrong, but opinion polls of large numbers of people who have every reason to know the facts can not be ignored. This survey was carried out among many informed users who include people that have worked in ICT for two or more years and the following are the responses tallied to average for each software they have used over the years.

4.2 Reliability

Table 4.1: Reliability of FOSS as compared to proprietary software

OSSF/FS	Rating (%)	Proprietary Software	Rating (%)
GNU Linux (server)	85	Windows 2003 server	75
GNULinux (client)	70	Windows XP	75
Apache Web Server	63	IIS	63
Mozilla Firefox	85	Internet Explorer	65

4.3 Software

In security, OSS/FS is the clear winner with better security features preferred by the user as the researcher discovered. Proprietary only managed to as good as FOSS on the Web server. Proprietary security features have improved as observed in the new Windows releases like Windows Vista. However, OSS/FS have also improved and can be customized to the user's liking, which is one of the reasons it was still the most preferred among the users.

4.4 Performance

Table 4.2: Performance of FOSS as compared to proprietary software

<i>OSSIFS</i>	Rating (%)	Proprietary Software	Rating (%)
GNULinux (Server)	85	Windows 2003 server	75
GNU Linux (Client)	65	Windows XP	75
Apache Web Server	70	uS	60
Mozilla Firefox	83	Internet Explorer	65

4.4 Results from the Experiment

4.4.1 Reliability Measure

A common reliability metric is the number of software faults, usually expressed as faults per thousand lines of code. The theory is that the software reliability increases as the number of faults (or fault density) goes down. In this research, reliability is usually measured by feeding programs with random characters and determining which programs resisted crashing and freeze-ups. This approach is unlikely to find subtle failures, but can still effectively find many errors in production software and is widely used for finding software errors. Most applications failed the reliability test with FOSS doing better than proprietary software as programs windows programs crashed more often. These programs crash due to error in code or non-reliability.

4.4.2 Performance

In performance testing that is performed, from one perspective, to determine how fast some aspect of a system performs under a particular workload. It can also serve to validate and verify other quality attributes of the system, such as scalability and reliability. Performance benchmarks are very sensitive to the assumptions and environment, so the best benchmark is one you set up yourself to model your intended environment.

When performance was measured by checking the speed of the software, FOSS was better than proprietary software. Internet Explorer (proprietary) was measured against Mozilla Firefox (FOSS) and we found out that web pages opened faster on Mozilla than Internet Explorer. Similarly, SuSe (FOSS) was much faster than Windows 2003 server (proprietary).

However, windows performed better in functions because it was more user friendly than its counterparts (GNU/Linux) and it could easily work with other software which made administration much easier.

4.4.3 Security Measure

Because security is hard to measure, it is often measured by a survey of opinions. As mentioned earlier, opinions can be wrong but when collected from a large sample space, how can these user's opinions be ignored? For this measure refer to the rating earlier reported in *section 4.1.3* above where FOSS was found to be better than FOSS.

When we applied a virus on Windows XP, it mal-functioned and had to be restored immediately before its ultimate crash. Windows Vista crashed immediately while Windows 2000 was affected by the virus but remained quite stable. These results are really bad because Linux only treated the virus as a program it was not compatible with. This goes to say how frustrating it is for Windows users having to update anti-virus after another. Its not that Linux is not affected by viruses but it is more immune to them.

4.5 The Case Study

Most organizations use both proprietary software and OSS/FS, while some use just one of them in their day-to-day operations. Internet presence for most organizations has posed a high degree of vulnerability to most of these organizations. UgandaDotNet was chosen as case study to this research. The case study findings are presented in a parallel manner with the results analysis that was done at the conclusion of this research. This allowed comparisons between the experimental and observational results.

4.5.1 Uganda Dot Net

Uganda Dot Net was launched in 2008. Its mission is to ensure the involvement of Ugandans in the development and use of open system software in Uganda, has been advocating and training of Ugandans in software development. The company focuses on

training users in the use of OSS and offer support as the companies call them to train their stuff. Though its worth noting that OSS has not taken off fully in Uganda. The organizations are involved in software development such as web site development and customized web applications, financial software corporate companies among others.

4.6 Results from Case Study Observation

The researcher observed that case studies were using FOSS and proprietary software. This included software like Windows XP and Vista (proprietary); as well as ZandrOS, Red Hat, Fedora and SuSe (FOSS) on the client side. On the server side, software used included Suse (FOSS) as well as Windows 2000 server and Windows 2003 Server. According to the findings, though both softwares were used, FOSS was placed at the core of the system because of its better reliability, security, and performance over proprietary software.

Also, for normal office operations, proprietary software was the most common sighting its being more user friendly to new or uninformed users. However simpler OSS/FS were competing favorably with proprietary with several users opting for Red Hat, Fedora, SuSe, XandrOS, Knoppix among others that had better security and reliability than open source.

Therefore, based on the observation, proprietary software was the preferred software program on the desktop given its easy of use. But closely followed by many *OSSIFS* products which in time will definitely out perform proprietary software. On the other hand, *OSSIFS* was the better software on the server-side given its better reliability and security features that can better protect a system from intruders and viruses.

4.7 Results from Interview Guide

Most people interviewed have been working in ICT for 2 years to 15 years and have used both *OSSIFS* such as GNU-Linux, Mozilla Firefox, Apache Web Server; and Proprietary Software such as Windows XP, Vista, Opera, IIS, Internet Explorer. Also interviewed were students, the public, and some company personnel that were the researcher assumed were either not very informed about ICT and software in general or they had spent a few days to a few months in ICT industry. However, it was noted that 90% of these users had purchased licenses for any of these software for a wide range of reasons. Organizations management were not willing to spend money on these licenses claiming they are expensive. The means of payment was also not easy as it has to be done via electronic payment which is also still virgin to the country. The process of acquisition for proprietary software is limited to a particular region which makes it expensive when one wants to add more computers. Most of these users intend to purchase these product licenses in future if for instance: The product corporations open offices in the country. If the prices for these licenses are lower. If the users can be allowed to freely distribute them beyond the specified regions as in the case of proprietary software.

The researcher also discovered that most users do not know have sufficient information about the benefits and challenges of running either licensed proprietary or Open Source software. Never the less, it was found out that if more information was provided to the users they would consider using the alternative software and even purchase the licenses. It was also found out that most informed users do use both OSS/FS and proprietary software. Proprietary software is used on desktop for users not familiar with OSS/FS while on the server-side they implemented the OSS/FS because of its reliability, performance and excellent security features. However, most of those that are not familiar with ICT all use only proprietary software because it is more user-friendly. This is due to lack of information on software benefits and skills that the public do not have. Unfortunately, the government is not supportive of the initiative of educating people on software use, and as was found out, any plans made have been shelved and await Non-

governmental Organizations (NGOs) companies like UGANDA DOT NET, among others to do this for them. This is frustrating for ICT because this is a big project that needs government support not just a few NGOs.

This chapter has analyzed interviews, observations, surveys and experiments carried out on the software performance, reliability and security. The chapter also described the results derived from the experiments and surveys and presented the information extracted from data collected during the study as well as the analysis derived from the findings.

CHAPTER FIVE

DISCUSSION, CONCLUSION AND RECOMMENDATIONS

5.0 Introduction

This chapter concludes the research report and covers discussions about the findings, conclusions drawn and offers recommendations of software to use for specific user requirements based on the findings of the study as well as the limitations and future areas of work related to this study.

5.2 Discussion

There are a lot of anecdotal stories that FOSS is more reliable, this research has provided quantitative and non-quantitative data confirming that mature FOSS programs are often more reliable.

Equivalent FOSS applications are more reliable than proprietary software according Fuzz report that shows evidence that Windows applications have even less reliability than the proprietary Unix software. Also, according total of 3 month Swiss evaluation, the difference between Netscape (proprietary) and Apache (FLOSS) is statistically insignificant.

IBM studies put Linux to a 10-month test and ran Caldera Systems OpenLinux, Red Hat Linux, and Windows NT Server 4.0 with Service Pack 3 on duplicate 100MHz Pentium systems with 64MB of memory found GNU/Linux highly reliable. The NT server crashed an average of once every six weeks yet none of the Linux server ever went down.

In addition to that, German import company Heinz Tröber found Linux-based desktops to be far more reliable than Windows desktops, where by Windows had a 15% daily failure rate, while Linux has 0%. On performance, FOSS has at least shown that it's often competitive, and in many circumstances it beats the competition. In 2002, TPC-C database measures found that a Linux based system was faster than a Windows 2000 based system. GNU/Linux with TUX has produced better SPEC values than Windows/IIIS in several cases, even when given inferior drive configurations.

GNU/Linux has better performance than Windows basing on their pipes (an input/output mechanism), process and thread creation, according to a study that examined the performance of pipes, a common low-level mechanism for communicating between program processes.

Microsoft themselves found that two OSS/FS operating systems, Linux and FreeBSD, had better performance than Windows by many measures according to a report that compares their research prototype to Windows, Linux, and FreeBSD exposing performance figures that compare these operating systems directly to each other.

On comparing security, a number of attempts to do so, suggest that FOSS is often superior to proprietary systems, at least in some cases. Unpatched Linux systems last longer than unpatched Windows systems according to a combination of studies from the Honeynet Project, AOL, Avantgarde and others and the Bugtraq vulnerability database suggests that the least vulnerable OS is OSSIFS. All the OSS/FS Oses in its study were less vulnerable than Windows in *1999-2000*. On responses to security problems, Red Hat (an OSSIFS vendor) responded more rapidly than Microsoft or Sun to advisories.

A 2002 survey of developers found that GNU/Linux systems are relatively immune from attacks from outsiders. Evans Data Corp.'s Spring 2002 Linux Developer Survey surveyed over 400 GNU/Linux developers, and found that Linux systems are relatively immune from attacks from outsiders. Apache has a better security record than Microsoft's IIS, as measured by reports of serious vulnerabilities by *Week's* July 20, 2001 article "Apache avoids most security woes" which examined that Apache's last serious security problem (one where remote attackers could run arbitrary code on the server) was announced in January 1997.

The majority of the most serious security problems only apply to Microsoft's products, and not to FOSS products as suggested by the CERT/CC's "most frequent, high-impact types of security incidents and vulnerabilities" and the ICAT database. Computer viruses are overwhelmingly more prevalent on Windows than any other system. According to a June 2004 study by Sandvine, 80% of all spam is sent by infected Windows PCs. National Cyber Security Alliance's study of May 2003 reported that 91% of Broadband users have spyware on their home computers running proprietary operating systems. In contrast, there's no evidence of that this is an issue for FOSS systems. National Cyber Security Alliance's study of May 2003 reported that 91% of Broadband users have spyware on their home computers running proprietary operating systems. In contrast, there's no evidence of this issue for OSS/FS systems.

Microsoft has had far more vulnerabilities than anyone else, according to Security Tracker Statistics paper (March 2002) that analyzed vulnerabilities from April 2001 through March 2002. In late June 2004, the U.S. Department of Homeland Security's Computer Emergency Readiness Team (CERT) recommended using browsers other than Microsoft Corp.'s Internet Explorer (IE) for security reasons because Microsoft had failed to patch a critical vulnerability for 9 months, and IE was being actively exploited in horrendous ways.

According to Symantec Corp., Mozilla Firefox fixed its vulnerabilities faster, and had fewer severe vulnerabilities (though more total vulnerabilities), in the July . December

2004 period than Internet Explorer. More recent summaries as of August 2005 suggest Internet Explorer is still more dangerous than the OSS/FS Firefox Security Fix reported that 78% (284/365) of the time in 2006 Internet Explorer was vulnerable to dangerous known attacks, for which no patch to fix it was available, compared to 2% (9/365) for Mozilla Firefox as reported by Brian Krebs's "Security Fix" column compiled statistics on vulnerability response times, including those for Microsoft

Internet Explorer (IE) and Mozilla Firefox. Internet Explorer (IE) users are far more likely to end up with a spyware-infected PC than Mozilla's Firefox users. Information Systems Journal (a peer-reviewed journal) published researcher Christian Payne's results, showing good evidence that OSS/FS can be secure. Information Systems Journal, Vol.12, Issue 1, February 2002, includes the peer-reviewed paper "On the security of open source software" by Christian Payne of Murdoch University (Perth, Australia).

A BZ Research survey of 6,344 software development managers shows Linux superior to Windows for operating system security attacks, and OSS/FS was in most categories considered equal or better at the application layer. Also, a BZ Research survey of 6,344 software development managers shows Linux superior to Windows for operating system security attacks, and OSS/FS was in most categories considered equal or better at the application layer: Therefore, based on the above discussions, it makes sense to select the most reliable product with the best security track record as well as better performance, even if no product has a perfect record.

5.3 Conclusion

The above observations are not merely opinions as these effects can be shown quantitatively, using a wide variety of measures. Using the measures above, it was discovered that Open Source Software is often the more reliable software, has better performance and better security, perhaps due to the possibility of worldwide review. Realizing these potential FOSS benefits may require approaching problems in a different

way. This will require an understanding of the differences between the proprietary and FOSS models. FOSS products are not the best technical choice in all cases, even organizations which strongly prefer FOSS generally have some sort of waiver process for proprietary programs.

Before deploying any program one needs to evaluate how well it meets the user's needs, Yet most organizations do not know how to evaluate software programs in general, specifically FOSS or proprietary software.

Therefore, it's worth noting that most organizations are switching to FOSS because cost is a significant factor driving adoption of open source software; control and flexibility are considered benefits; Implementation of open solutions is evolutionary, not revolutionary; Open source extends across the entire software stack and Product support is not a significant concern.

However, most companies maintain the use of both software (OSS/FS and Proprietary) because of they claim that though OSS/FS programs are more secure, reliable, perform better than proprietary products but they are not user friendly. Users prefer to use OSS/FS on the Server-side where security attacks are frequent and proprietary products on the desktop.

Also interfacing some programs on OSS/FS to proprietary programs like Ms Windows is quite frustrating for most users of proprietary products. So they prefer to exclusively use proprietary products. Many FOSS programs aren't available for Windows, or do not work as well on Windows.

5.4 Recommendations

Based on the findings of this study with an edge of performance, reliability and security. OSS/FS options should be carefully considered any time software or computer hardware

is needed. Organizations should ensure that their policies encourage, and not discourage, examining OSS/FS approaches when they need software.

Many organizations are opting to computerize their operations to cope with the competition in the business market by establishing ICT departments. To achieve their objectives, special care and attention must be taken when purchasing software by

Objectively with knowledge about functionality of the software choose the best software to do the job effectively. Also, the government should endeavor to support ICT training especially in OSS/FS so people can have a wider choice of products to improve on their productivity and compete favorably not only within the country but world-wide since its increasingly becoming a global village.

Those interested in trying out GNU/Linux operating system often start with a simple CD that doesn't touch their hard drive, such as Gnoppix or Knoppix, then move on to various Linux distributions such as Red Hat (inexpensive Fedora Core or professionally- supported Red Hat Enterprise Linux), Novell/SuSE, Mandriva (formerly MandrakeSoft), or Ubuntu (nontechnical users may also be interested in pay-per-month distributions like Linspire, while technically knowledgeable users may be interested in distributions like Debian).

Users of Windows programs who are looking for software often try desktop programs such as OpenOffice.org (OSS/FS office suite), Firefox (OSS/FS web browser), and Thunderbird (*OSSIFS* mail browser). The OpenCD project creates CDs that include those (and other) *OSSIFS* programs and Windows. It must be noted that the software and standards that drive these ICTs into the future can shape the country's economy and even go a long way to influence who is included or excluded. Therefore the choice of *OSSIFS* and proprietary software

is an issue whose importance goes far beyond the boundaries of the software industry alone.

5.5 Areas for Future Studies

According to our findings, most users of proprietary software are hope to move to OSS/FS or use both if more information was availed them. Also, there are several reports from various users who have switched to FOSS from proprietary software and a few who have moved to proprietary to FOSS. These moves often occur as a result of user's requirements and needs. The reason for the move from FOSS to proprietary is that, users feel that FOSS is not user friendly and does not work well with some proprietary software.

However, some users preferred to use both software in time. Most OSS/FOSS is being made as user friendly as possible such that any one will be able to use without prior training. UGANDA DOT NET is not taking any chances though as it aims at making sure as many people get to know more about OSS/FOSS such that users can make an informed choice when purchasing software. However, more work is needed in this area to train users in the operation of this software. Educational organizations have found OSS/FOSS software useful. KIU has moved to more extended use of OSS/FOSS. Its hoped that more *OSS/FOSS* installations will be implemented as they found out that the Linux systems are faster than their machines enabling them to produce much higher quality results. They also use Python extensively (an OSS/FS language), as well as a number of in-house and proprietary tools.

2. Wikipedia. Free Encyclopedia. (2007, October) Open-source software
Website: [http://en.wikipedia.org/wiki/Open-source software](http://en.wikipedia.org/wiki/Open-source_software)

3. Netcraft. (2007). *Netcraft Secure Server Suivey*.

Website: <http://uptime.netcraft.com/>

4. Richard Gooch. (12 Nov 2006). *The linux-kernel mailing list FAQ*.
Website: <http://www.tux.org/lkm> 11

5. Standard Performance Evaluation Corporation. (2007). <http://www.spec.org/>

6. Week. Linux Watch. (October, 2007). *13 reasons why Linux should be on your desktop*.

Website: <http://www.eweek.com/article2/>