Security of Open Source and Closed Source Software :An Empirical Review of Published Vulnerabilities

Mr. Selassie Kennedy Kofitse
Research Scholar
Sikkim Manipal University,
Accra, Ghana
selaseken@gmail.com

ABSTRACT: ***Literature on open source and closed source security most often than not is based on the posture of the discussants, be it pro open source or close source which is often determined by biased attitudes toward one of these development styles. The discussion is normally driven by emotions instead of hard data. To bridge the opinionated attitude towards this subject this paper contributes to solving this problem by analyzing and comparing published vulnerabilities of six open source software and six closed source software packages, all of which are widely deployed. Thereby, it provides an extensive empirical analysis of vulnerabilities in terms of mean time between vulnerability disclosures, and the severity of vulnerabilities. The investigation reveals that (a) the mean time between vulnerability disclosures was lower for open source software studied, (b) regarding the severity of vulnerabilities, no significant differences were found between open source and closed source vulnerabilities (c) it was established that "Given enough eyeballs, bugs are shallow." concept holds true for open sources since more vulnerabilities are identified and fixed within records time too.***

KEYWORDS: ***Vulnerabilities, security, open source software, closed source software***

I. INTRODUCTION

The logic is understandable - how can software with source code that can easily be viewed, accessed and changed have even a modicum of security? In the past, software is acquired by procuring licenses for a proprietary, or binary-only, immaterial "object". Then software was regarded as a good that we have to pay for just as we would pay for material objects, such as car or food. In contemporary times, this widely cultivated habit has begun to be accompanied by a new model, which is characterized by software that comes with a compliable source code. The source code is made available, free of charge, to all interested parties; further users have the right to modify and extend the program in most cases these days. Open Source Software (OSS) methods rely on developers who reveal the source code under an open source license. Historically, much of the software was completely available without a precise license governing its use. This liberal state of affairs led to unethical use and behaviour of programmers. This led to the development of new approaches with reference to distribution of software by Richard Stallman, a programmer at Massachusetts Institute of Technology in the 1980s, where he suggested that the license should be General Public License (GNU). Under certain types of open source licenses, any further development using the source code must also be publicly disclosed. In discussing open source [9] reminds us that, free and open source software dates right back to the origins of the computing field, as far back as the 1950s, when all software were free, and most of it open[6].

While there is consensus that opening source code to the public increases the potential number of reviewers, its impact on finding security flaws is controversially debated. Proponents of open source software stress the strength of the resulting review process and argue in the sense of [7] that, *"Given enough eyeballs, bugs are shallow.",* while some opponents follow the argument of [5], who remarks *"Sure, the source code is available. But is anyone reading it?"* Interestingly, both parties essentially agree that open source basically makes it easy to find vulnerabilities; they only differ in their conclusions with regard to the resulting impact on security[6].

In order to have an unbiased discussion on open source and closed source security, it is helpful, if not necessary, to transparently measure the empirical security of software – be it open source or closed source software. However, measuring security is a challenging task, because security is somehow invisible. Despite an increasing number of quantitative research papers on measuring software security in the past years, it is still true what[11] observed: what the discussion on software security specifically lacks is appropriate metrics, methodology and hard data.

Addressing this research gap, this thesis analyzed and compares published vulnerabilities of six open source software and six closed source software packages, all of which are widely deployed. More specifically, this empirical study statistically analyses vulnerabilities in terms of the mean time between vulnerability disclosures, and the severity of vulnerabilities.

II. RESEARCH REVIEW/METHODLOGY

A. Open and closed source software

The availability of source code to the public is a precondition for software being denoted as "open source software" in most cases. Beyond this requirement, the Open Source Initiative (OSI) has defined a set of criteria that software has to comply with[10]. The definition particularly includes permission to modify the code and to redistribute it. However, it does not govern the software development process in terms of who is eligible to modify the original version. When what is called "bazaar style" by [5] is in place, any volunteer can provide source code submissions. Software development is then often based on informal communication between the coders[10]. In a more closed environment, software is crafted by individual wizards and the development process is characterized by a relatively strong control on design and implementation. This style is referred to as "cathedral style" [5]. The implementation of this modification procedure might have an impact on the security of software, so that a detailed discussion of open source security would need to consider it.

Open source software is widely held to be more secure than closed source software. The core of the argument is that with open source code, many people have the potential to find and correct an error. This is summarized as "given enough eyeballs, all bugs are shallow" [10]. While researchers have attempted to quantify and measure this effect [11], it is inherently complex. Software projects differ in complexity, features, scope, and user base; the number and severity of vulnerabilities may be linked to these differences. Therefore, attributing the vulnerabilities to the open/closed choice is difficult. Furthermore, recent research suggests that there may be diminishing returns to increased number of users in the context of software or other community build artifacts[10].

Therefore, many closed source projects could already have "enough eyeballs" and open source projects could have more than enough. In fact, recent empirical research finds limited differences in vulnerabilities disclosed in each type, but also finds some evidence of more frequent disclosures in open source [11].

B. Vulnerabilities

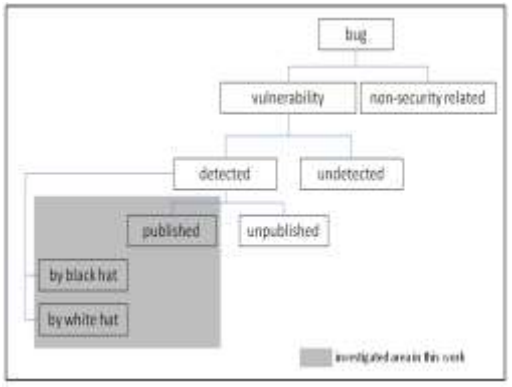

Figure 1. Classification of software bugs and vulnerabilities

A software bug is an error, flaw, failure, or fault in a computer program or system that causes it to produce an incorrect or unexpected result, or to behave in unintended ways. Most bugs arise from mistakes and errors made by people in either a program's source code or its design, or in frameworks and operating systems used by such programs, and a few are caused by compilers producing incorrect code.

Bugs trigger errors that can in turn have a wide variety of ripple effects, with varying levels of inconvenience to the user of the program. Some bugs have only a subtle effect on the program's functionality, and may thus lie undetected for a long time. More serious bugs may cause the program to crash or freeze. Others qualify as security bugs and might for example enable a malicious user to bypass access controls in order to obtain unauthorized privileges.

When software is executed in a way different from what the original software designers intended, this misbehaviour is rooted in software bugs. . The portion of bugs that are security- critical ("vulnerabilities") is assumed to be 1% [2], resulting to an amazingly high figure of 350,000 vulnerabilities in Windows 2000. Detected vulnerabilities can further be divided into those being published and unpublished.

Vulnerabilities are (software) product-related weaknesses, for which publicly accessible databases are available. Rooted in these are concrete security incidents (breaches), which are system-related and cause the actual harm. Breaches are much more difficult to investigate, because data is scarcer.

Once vulnerability is detected, the question arises whether to disclose it or not. Researcher argues against disclosure unless vulnerabilities are correlated. However, investigating the operating system FreeBSD finds vulnerabilities being correlated regarding its' rediscovery and argues in favour of disclosure. Using game -theoretic models, address the question of when

software vulnerabilities should be disclosed and conclude that neither instant disclosure nor non-disclosure is optimal[3][4].

C. Software Packages and Data Sources

Is there a difference between the publication of vulnerabilities occurring in open source and closed source applications?

Ho: There is no significance difference between vulnerabilities disclosure of open source and close source application.

H1: There is a difference between vulnerability disclosure for open source and close source.

The selection of software packages to get investigated is driven by the goals to:

i. have open and closed source software systems that serve the same purpose (for the sake of comparability),
ii. consider software that is known and relevant to the community.

Each of the selected software bundles is analyzed regarding its vulnerabilities, as published in the National Vulnerability Database (NVD) of the National Institute of Standards and Technology (NIST). This database is one of the most comprehensive vulnerability databases. I analyze each software product regarding the number of vulnerabilities, the disclosure rate, and the severity of vulnerabilities.

Application Type	Product
Browsers	Internet Explorer 7
	Mozilla Firefox 3.0
Email Client	Ms Outlook 12
	Thunderbird 3.0
Web Server	IIS 4.0
	Apache 2.2
Office	MS Office 2007
	Open Office 3
Operating Systems	Windows 7
	Red Hat Linux 6
Database Management Systems	PostgreSQL 9
	Oracle 11g

Table1: Bundle of applications for the study.

D. Vulnerability Measurement

In measuring vulnerability "mean time between vulnerability disclosures" (MTBVD) was defined as the number of days since a software is release divided by the number of published vulnerabilities. With regard to determining the MTBVD, consideration was given to only those vulnerabilities that have been published after the release date and limited between January 2010 and February 2015. This does not include vulnerabilities before the current release version of the software in question.

A simple comparison of MTBVD is not assumed to provide reliable results regarding the level of security, because vulnerability detection and publication alone could be influenced by other factors too.

III. DATA COLLECTION

In the few empirical studies on software security [1] [7][8], the application types mainly considered are operating systems, web browsers, web servers, email clients, and database management systems. Adopting this focus, this study considers two operating systems (Windows 7, Red Hat Enterprise Linux 6), two web browsers (Internet Explorer 7, Mozilla Firefox 3), two web servers (IIS 6, Apache 3), two email clients (MS Outlook 12, Thunderbird 3), two database management systems (PostgreSQL 9, Oracle 11g), and two office products (MS Office 2007, Open Office 3).

A. Vulnerability sources

I consider those vulnerabilities that have been accepted as Common Vulnerabilities and Exposures (CVE) by MITRE. Each of these vulnerabilities has a unique identifier, e.g. CVE-2015-0836. CVE identifiers are also used as references in many other vulnerability databases. The NVD feeds contain data on the severity and type of vulnerabilities. Data excludes misconfigurations (CCE = Common Configuration Enumeration).

Overall, I consider two types of vulnerabilities: those that are explicitly applicable to the software version under consideration, and those that affect all versions of the particular software and that have been published after the release date of the considered version. The data used in this work refer to vulnerabilities that have been published prior to 28 February 2015 and after 01 January 2010.

The National Vulnerability Database is the U.S. government repository of standards-based vulnerability management data represented using the Security Content Automation Protocol (SCAP). This data enables automation of vulnerability management, security measurement, and compliance. NVD includes databases of security checklists, security related software flaws, misconfigurations, product names, and impact metrics. NVD supports the Information Security Automation Program (ISAP). In addition to providing a list of Common Vulnerabilities and Exposures (CVEs), the NVD scores CVEs to quantify the risk of vulnerabilities, calculated from a set of equations based on metrics such as access complexity and availability of a remedy.

IV. RESULT ANALYSIS

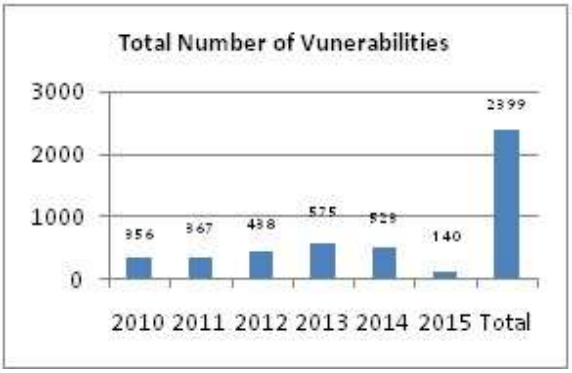

Figure 2: Vulnerability Disclosures by Years

	Apache2.2	IIS 6.0	Internet Explorer	Mozilla Firefox	Office 2007	Open Office	Oracle 11	Microsoft Outlook 12	Thunderbird	Postgre SQL 9	Red Hat Linux 6	Windows 7	Grand Total
2010	4	6	2	106	55	8	33	5	62	7	4	64	356
2011	7	0	0	98	30	9	49	0	65	1	6	102	367
2012	2	1	18	162	19	6	26	0	147	9	4	44	438
2013	4	1	129	149	17	4	14	4	113	6	34	100	575
2014	0	1	243	108	10	2	41	5	64	9	4	36	523
2015	0	0	52	28	5	0	7	4	8	0	1	35	140
Grand Total	17	9	444	651	136	29	170	18	459	32	53	381	2399

Table 2: Vulnerability disclosure by individual applications by years.

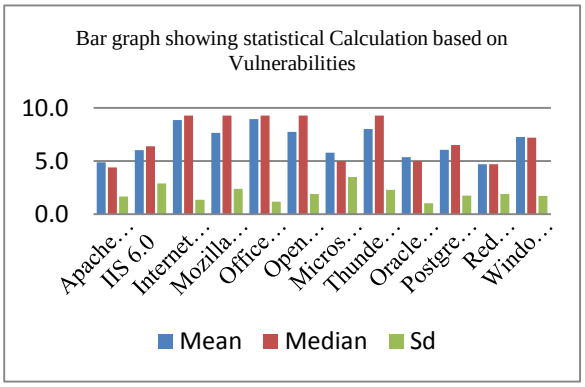

Figure 4: Showing Statistic of the applications

Application	Number of Vulnerabilities	Released Year	Number of Years	Number of Days	MTBVD	Current Usage/Market Share
IIS 6.0	9	2007	8	2921.94	324	13.2***
Apache2.2	17	2005	10	3652.42	214.4	58.5***
Microsoft Outlook 12	18	2007	8	2921.94	162.3	******5.0
Thunderbird 3.0	459	2009	5	1821.21	4.0	No data
Office 2007	136	2007	8	2921.94	21.5	72.00*****
Open Office 3	29	2008	6	2556.7	88.2	21.2*****
Red Hat Linux 6.0	53	2010	4	1460.97	27.6	1.53*
Windows 7	381	2009	5	1826.21	4.8	55.99*
Oracle 11g	170	2007	8	2921.94	17.2	32.8****
PostgreSQL 9.0	32	2010	4	1460.97	45.7	5.6****
Internet Explorer 7	444	2006	9	3287.18	7	57.3**
Mozilla Firefox 3.0	651	2008	6	2556.7	18.8	11.6**

Table 3: MTBVD and market shares

Bundles	Application	Low	Medium	High	Grand Total	Percentage Severely high
Web Servers	Apache2.2	2	12	3	17	17.6
	IIS 6.0	3	5	1	9	11.1
Browsers	Internet Explorer 7	12	426	6	444	1.4
	Mozilla Firefox 3.0	309	318	24	651	3.7
Office	Office 2007	3	133	0	136	0.0
	Open Office 3	3	26	0	29	0.0
Email Client	Microsoft Outlook 12	1	11	6	18	33.3
	Thunderbird 3.0	232	217	10	459	2.2
Database Management System	Oracle 11g	95	46	29	170	17.1
	PostgreSQL 9.0	21	11	0	32	0.0
Operating Systems	Red Hat Linux 6.0	18	30	5	53	9.4
	Windows 7	242	131	8	381	2.1
	Total	**941**	**1366**	**92**	**2399**	3.8

Table 4: Bundle of applications and levels of vulnerability.

V. DISCUSSIONS

A total of 2399 vulnerabilities were published between the defined dates for the selected bundles of study. 2013 recorded the highest made up of (578) 24 percent of total vulnerabilities across the various years and 2015 recorded (140) 5.8 percent.

A total of 1241 constituting 51.7 percent of all vulnerabilities for the study period was open source while 1158 constituting 48.3 was closed source for the subjects under study. This shows that software's performing the same task and of the widely used open source vulnerabilities are to be found faster and patched than that of close source.

Web Servers: Apache 2.2 been open source web servers have 17 recorded vulnerabilities with 17.6 percent been severely high while Internet Information Services (IIS 6.0) closed source recorded 9 vulnerabilities with 11.1 percent.

The web server subjects in this study shows that open source web servers have more vulnerabilities and their level of severity is more than that of close source. This can also be due to the fact that the usage of the open

source is more wide spread than that of the close source web servers.

Browsers: Mozilla Firefox 3.0 open source have 651 vulnerabilities with 3.7 percent been severely high and 48.8 percent medium severe while Internet Explorer 7.0 closed source have a total of 444 vulnerabilities with 1.4 percent been severely high and 96 percent falling in the medium severity bracket.

The concept of whether more vulnerability will be disclose in open source compared to close source is established by the facts from the browsers selected in this study. It is also instructive to note that the level of the severity of the vulnerabilities in open source is more than that of close source.

Office: Open Office 3.0 open source had 29 vulnerabilities recorded for the study period and no severity while Microsoft Office 7 closed source had 136 vulnerabilities with no severity vulnerabilities. Most of the vulnerabilities for these applications were found within the medium bracket.

The level of vulnerability disclosure in the office applications selected in this study, shows that close source have less vulnerabilities disclose while close source have more vulnerability. It is also reveling that despite the fact of close source office have more vulnerabilities compared to open source, none of them recorded high severity vulnerabilities. This is highly significant.

Email Client: Thunderbird 3.0 open source with 459 vulnerabilities and 2.2 percent severity vulnerabilities while Microsoft Outlook 12 close source have 18 vulnerabilities with 33.3 percent level of severity.

The subjects selected under email client results are very interesting. The open source thunderbird with as high as 459 disclosed vulnerabilities have only 2.2 percent severity rate compared to 33.3 percent of high severity rate for 18 vulnerabilities disclosed for close source. This presupposes that the more vulnerability likely to be discovered for the close source software will put it in a very high risk area. The assertion that more vulnerabilities will be disclose in open source still stands.

Database Management System: PostgreSQL 9.0 open source with a total of 32 vulnerabilities and no level of severities while Oracle 11g close source has 170 vulnerabilities and 17.1 percent level of severity.

The database management systems subjects' results are reveling. PostgreSQL have thirty-two vulnerability without any high severity vulnerabilities while oracle has as much as 170 published vulnerabilities with 17.1 percent high vulnerabilities. In conclusion based on the results database management systems close source is much susceptible to vulnerability disclosure than that of open source.

Operating Systems: Red Hat Linux 6.0 open source with a total of 53 vulnerabilities and 9.7 percent level of severity while Windows 7 had 381 vulnerabilities with 2.1 percent level of severity.

The vulnerability disclosure in the operating systems shows that despite the fact that less vulnerability were found in open source operating systems there is high percentage severity of these vulnerabilities compared to the close source which have high number of vulnerability disclosure with less percentage of high severity vulnerability.

In conclusion only three open source applications by the bundles which meets the assertion that open source software are more likely to have more vulnerability disclosures than close source. They are Apache, Mozilla Firefox and Thunderbird in the web servers, Email client and browsers bundles.

Apache 2.2 which is open source software recorded the highest number of days (214) for MTBVD for the set of subjects studied. This is based on the backdrop of 58.5 percent market share or usage as at February, 2015 (W3Tech 2015). In the same perspective Thunderbird another open source recorded the least number of days (4) before the first vulnerabilities was disclosed. It is worth nothing that the other open source software's namely, Mozilla Firefox 3.0, Open Office 3.0, PostgreSQL 9.0 and Red Hat Linux 6.0 has 19, 88, 17 and 28 MTBVD respectively.

However, Internet Information Services (IIS 6.0) close source recorded the highest number of days (324) for MTBVD while Windows 7 another close source software have its first vulnerabilities disclosed within 7 days.

Juxtaposing the MTBVD with the usage or market share of the application, it is not far fetch to categorically state that the more attractive or the usage of the application the likely hood of finding a very low MTBVD. But the data available does not support this assertion. Therefore I conclude that vulnerability disclosure have no direct relationship between market share and methodology of developing the application.

The classification of severity of vulnerability used for the study was based on what is used in the NVD which is the data source for the study.

1. Vulnerabilities are labeled "Low" severity if they have a CVSS base score of 0.0-3.9.

2. Vulnerabilities will be labeled "Medium" severity if they have a base CVSS score of 4.0-6.9.

3. Vulnerabilities will be labeled "High" severity if they have a CVSS base score of 7.0-10.0.
The medians of medians reveal that the vulnerabilities of office products and Browser and are much more severe (9.3) than those of Email clients (7.2), while the values of the other application types are close to each other. The empirical analysis shows differences in terms of vulnerability severity for different application types. However, the number of investigated software bundles is still too low to deduce general hypotheses. An investigation of the type of vulnerabilities might reveal the reasons for the observed differences.

When we determine the medians of medians of open source software (6.5) and closed source software (7.0) and also the corresponding medians of the percentage of highly severe vulnerabilities (3.4 and 4.3, respectively), the first impression is that open source software is more secure in terms of the level of severity. Summing up, I find no significant difference between the severity of vulnerabilities in open source and closed source software. This conclusion is based on the two tailed t test statistics of P = 0.127 which is not significant.

VI. CONCLUSION AND FUTURE WORKS

The study was conducted to find an empirical data to establish the whether there is a difference between vulnerability disclosure in open source or close source application. The fact shows that a total of 1241 constituting 51.7 percent of all vulnerabilities for the study period ware open source while 1158 constituting 48.3 was closed source for the subjects under study. On the face of this fact it is easy to easily conclude that disclosure of vulnerability in open source is higher than close source, but the sample for the study is not that large so it cannot be generalized. In addition only three open source applications by the bundles which meets the assertion that open source software are more likely to have more vulnerability disclosures than close source. They are Apache, Mozilla Firefox and Thunderbird in the web servers, Email client and browsers bundles.

The is the need to do in-depth research into areas of particular vulnerability for open source and close source, Vulnerability development over time and software line of codes to determine their confounding factors on vulnerability development and disclosure.

VI. REFERENCES

[1] Alhazmi, O., Malaiya, Y., & Ray, I. (2007). *Measuring, analyzing and predicting security vulnerabilities in software systems*: Computers & Security, 26, 3, 219-228.

[2] Anderson, R. (2001). *Why Information Security is Hard - An Economic Perspective*, in Proceedings of the Seventeenth Computer Security Applications Conference: New Orleans, December 10-14, 358-365.

[3] Arora, A., Krishnan, R., Nandkumar, A., Telang, R., & Yang, Y. (2004). *Impact of Vulnerability Disclosure and Patch Availability* - An Empirical Analysis, in Proceedings of the Third Workshop on the Economics of Information Security: University of Minnesota, May 13-14, 1-20.

[4] Arora, A., Telang, A., & Xu, H. (2004). *"Optimal Policy for Software Vulnerability Disclosure*", in Proceedings of the Third Annual Workshop on Economics and Information Security: University of Minnesota, May 13-14, 52-59.

[5] Christensen, Mark J., & Richard H. Thayer(2001). *The Project Manager's Guide to Software Engineering's Best Practices*. Los Alamitos: IEEE.

[6] Costa, D., & Scott, V. (2005). "CNET editors' review". CNET Reviews.

[7] Hiran, K.K., & Doshi, R.(2014). "The Proliferation of Smart Devices on Mobile Cloud Computing" , LAMBERT Academic Publishing ISBN: 3659573914

[8] Hiran, K. K., Doshi, R. & Rathi, R.(2014). *"Security & Privacy Issues of Cloud & Grid Computing Networks"*, International Journal on Computational Sciences & Applications,4(1), 83-91.

[9] Schechter, S.(2004). *Toward Econometric Models of the Security Risk from Remote Attacks.* Workshop on the Economics of Information Security.

[10] Tute, C. (2014). *Handling Security Issues In Open Source Projects;* Retrieved 12 24, 2014 from, https://robots.thoughtbot.com/handling-security-issues-in-open-source-projects.

[11] Witten, B., Landwehr, C., & Caloyannidis, M. (2001). *Does open source improve system security?*, in IEEE Software,18,5, 57-61.