Open Source vs. Proprietary Software: Vulnerabilities and Patch Response  
Sanjay Sridhar, Kemal Altinkemer, and Jackie Rees

Overview

- Introduction
- Hypotheses
- Data
- Preliminary Results
- Conclusions and Future Research

Research Questions

- Are open source developers quicker to release patches than proprietary S/W developers?
- Does open source S/W have fewer vulnerabilities than proprietary S/W?
- Does open source S/W have more severe vulnerabilities than proprietary S/W?
- Are “confidentiality-threat” vulnerabilities more common in open source than proprietary S/W?

Introduction

- Software (S/W) vulnerabilities represent targets for misuse or error
- Software vulnerabilities as software reliability issue
- Patches are released as “solution” to vulnerability
- New S/W releases incorporate previous patches
- Are there differences between patching activities of open source SW developers and proprietary SW developers?

Hypotheses

- Hypothesis 1: Open source software developers issue patches faster than proprietary software vendors.
- Hypothesis 2: In unit time, there are fewer vulnerabilities in open source software compared to proprietary software.
- Hypothesis 3: Open source software patches high-severity vulnerability faster than proprietary software.
- Hypothesis 4a: Open source software developers patch confidentiality-type vulnerabilities faster than proprietary software developers.
- Hypothesis 4b: Open source software developers patch integrity-type vulnerabilities faster than proprietary software developers.
- Hypothesis 4c: Open source software developers patch availability-type vulnerabilities faster than proprietary software developers.
- Hypothesis 4d: Open source software developers patch security protection-type vulnerabilities faster than proprietary software developers.

Data & Methodology

- Operating System Vulnerability and Patch data from the Common Vulnerabilities and Exposures Database (http://www.cve.mitre.org)
- Data collected includes:
  - Vulnerability description
  - CVE key code
  - Date of discovery
  - Date of patch response

Table 1. Mean time to patch for open-source & proprietary operating system software. 1-tailed p-values at the 10% significance shown in bold.

<table>
<thead>
<tr>
<th>Year</th>
<th>Open Source Software</th>
<th>Proprietary Software</th>
<th>T-stat</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>N Mean (days) Std. Dev.</td>
<td>N Mean (days) Std. Dev.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td>37 57.76  57.52</td>
<td>17 39.19  23.00</td>
<td>2.06(0.03)</td>
<td></td>
</tr>
<tr>
<td>2001</td>
<td>94 30.64  41.77</td>
<td>83 37.03  47.90</td>
<td>1.02(0.16)</td>
<td></td>
</tr>
<tr>
<td>2002</td>
<td>81 34.12  37.78</td>
<td>105 42.66  47.49</td>
<td>1.41(0.18)</td>
<td></td>
</tr>
<tr>
<td>2003</td>
<td>77 29.68  48.26</td>
<td>84 41.12  54.99</td>
<td>1.30(0.18)</td>
<td></td>
</tr>
<tr>
<td>2004</td>
<td>17 24.73  14.75</td>
<td>27 24.41  18.50</td>
<td>0.00(0.40)</td>
<td></td>
</tr>
<tr>
<td>2005</td>
<td>305 34.30  43.81</td>
<td>317 38.90  47.00</td>
<td>-1.42(0.15)</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 1. Inter-arrival times of vulnerabilities for open-source & proprietary operating system software.
Open Source vs. Proprietary Software: Vulnerabilities and Patch Response
Sanjay Sridhar, Kemal Altinkemer, and Jackie Rees

1.27 (0.12) 31.51 28.07 15 27.64 42.91 11
Low

-0.50 (0.31) 41.90 36.11 131 39.53 33.64 146
Medium

-1.33 (0.09) 51.98 41.99 171 51.30 34.30 148
High

Table 3. Severity of the vulnerability. The 1-tailed p-values at the 10% significance are shown in bold.

Significant difference in performance regarding high-severity vulnerabilities
H3 is supported

Table 4. Type of vulnerability. The 1-tailed p-values at the 10% significance are shown in bold.

Significant difference in performance regarding high-severity vulnerabilities
H3 is supported

No significant difference among types of vulnerabilities
H4a, H4b, H4c, and H4d not supported

Conclusion & Future Research

• Open-source software appear to be more reliable.
• Develop a software-reliability model.
• Other parameters to evaluate performance (E.g. Number of programmers, maturity of the application)